공학박사 학위논문

Energy Awareness for Multiple Network Interface-Activated Smartphone

다중 네트워크 인터페이스가 활성화된 스마트폰을 위한 에너지 인지 기법

2017년 8월

서울대학교 대학원

전기·컴퓨터공학부

구 종 회

Abstract

State-of-the-art smartphones can utilize multiple network interfaces simultaneously, e.g., LTE and Wi-Fi, to enhance throughput and network connectivity in various use cases. In this situation, energy consumption of smartphones can increase while using both LTE and Wi-Fi interfaces simultaneously. Since energy consumption is an important issue for smartphones powered by batteries with limited capacity, it is required to consider the trade-off between energy increase and performance enhancement. By judiciously utilizing both LTE and Wi-Fi interfaces of smartphones and energy awareness techniques, it is enabled to optimize the performance of smartphone's applications while saving battery energy.

In this dissertation, we consider the following three strategies to enable the energy awareness for smartphones which utilize both LTE and Wi-Fi links: (i) Power modeling for smartphone which utilizes both LTE and Wi-Fi links simultaneously, (ii) real-time battery drain rate estimation for smartphones, and (iii) optimizing the performance of dynamic adaptive streaming over HTTP (DASH)-based video streaming for smartphones.

First, an accurate power modeling is presented for the smartphones, especially, those capable of activating/utilizing multiple networks simultaneously. By decomposing packet processing power and power consumed by network interfaces, we construct the accurate power model for multiple network interface-activated cases. The accuracy of our model is comparatively evaluated by comparing the estimated power with the measured power in various scenarios. We find that our model reduces estimation error by 7%–35% even for single network transmissions, and by 25% for multiple network transmissions compared with existing power models.

Second, in order to enable real-time energy awareness for smartphones, we develop a battery drain rate monitoring technique by considering characteristics of Li-ion bat-

teries which are used by smartphones. With Li-ion battery, battery drain rate varies with temperature and battery aging, since they affect battery characteristics such as capacity and internal resistance. Since it is difficult to model the battery characteristics, we develop *BattTracker*, an algorithm to estimate battery drain rate without knowing the exact capacity and internal resistance by incorporating the concept of effective resistance. *BattTracker* tracks the instantaneous battery drain rate with up to 0.5 second time granularity. Extensive evaluation with smartphones demonstrates that *BattTracker* accurately estimates the battery drain rate with less than 5% estimation error, thus enabling energy-aware operation of smartphones with fine-grained time granularity.

Finally, we adapt an energy awareness and utilize both LTE and Wi-Fi links for a Dynamic Adaptive Streaming over HTTP (DASH)-based video streaming application. Exploiting both LTE and Wi-Fi links simultaneously enhances the performance of DASH-based video streaming in various aspects. However, it is challenging to achieve seamless and high quality video while saving battery energy and LTE data usage to prolong the usage time of a smartphone. Thus, we propose REQUEST, a video chunk request policy for DASH in a smartphone, which can utilize both LTE and Wi-Fi to enhance user's Quality of Experience (QoE). REQUEST enables seamless DASH video streaming with near optimal video quality under given budgets of battery energy and LTE data usage. Through extensive simulation and measurement in a real environment, we demonstrate that REQUEST significantly outperforms other existing schemes in terms of average video bitrate, rebuffering, and resource waste.

In summary, we propose a power modeling methodology, a real-time battery drain rate estimation method, and performance optimization of DASH-based video streaming for a smartphone which utilizes both LTE and Wi-Fi simultaneously. Through this research, we propose several energy-aware techniques for the smartphone, which especially utilizes both LTE and Wi-Fi, based on prototype implementation and the real measurement with experimental equipment. The performance of the proposed meth-

ods are validated by implementation on off-the-shelf smartphones and evaluations in real environments.

keywords: Wi-Fi, LTE, smartphone, power modeling, Li-ion battery, energy awareness, dynamic adaptive streaming over HTTP (DASH).

student number: 2011-20785

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Smartphones can activate both Wi-Fi and LTE simultaneously to achieve higher throughput or to maintain seamless connection to a server. In this case, the energy consumption of a smartphone varies depending on the number of activated network interfaces and their usage. Meanwhile, measuring the device power consumption using a power meter to evaluate the energy efficiency of a certain algorithm is a fairly burdensome job, thus motivating researchers to develop power consumption models. However, the existing power models are limited to single network-activated cases. Thus, a new power model should be developed to understand how much energy is consumed for each network interface and how much energy can be saved by utilizing both LTE and Wi-Fi links.

Even if devices consume the same amount of power, battery drain rate can vary according to their battery capacity. Furthermore, battery characteristics such as capacity and internal resistance vary according to temperature and the degree of battery aging [1, 2]. For these reasons, even though the existing power modeling-based methods accurately estimate energy consumption, they fail to provide accurate battery drain rate in varying temperature or for a device powered by an aged battery. Therefore, it is necessary to predict the battery drain rate in real-time to provide a user with an

expected battery lifetime. Being motivated by this, to realize energy-aware technologies for smartphones, we develop a real-time battery drain rate estimation algorithm in addition to a power consumption model.

A good example of how to optimize performance by using multiple network interfaces, smartphone power modeling, and battery drain rate estimation techniques is video streaming. However, considering realistic situations such as mobile environments, there are many issues to optimize the performance of video streaming. While a mobile device can utilize both Wi-Fi and LTE networks simultaneously, the Wi-Fi link may become unstable and its quality may fluctuate severely especially in a mobile or dense environment. For example, when a user moves around and goes out of the coverage of a Wi-Fi AP connected to the user's mobile device, the device cannot utilize Wi-Fi link until it finds an available Wi-Fi AP nearby and handover to a new Wi-Fi AP is successfully completed. It is also well known that Wi-Fi throughput degradation occurs in mobile environments due to poor performance of handover operations in commercial Wi-Fi devices [3–5]. In addition, if a Wi-Fi AP operates at 2.4 GHz, the Wi-Fi link quality may severely suffer from interference caused by other wireless devices operating in 2.4 GHz ISM band, such as Bluetooth, ZigBee, microwave ovens, and cordless phones [6, 7]. Furthermore, Wi-Fi at 5 GHz suffers from higher path loss than Wi-Fi at 2.4 GHz, thus increasing the possibility that mobile user experiences worse Wi-Fi link quality and goes out of Wi-Fi coverage. In contrast to Wi-Fi, a device may retain a seamless connection via the LTE network even though user moves fast, thanks to seamless handover between LTE base stations.[1] Likewise, in mobile and dense environments, Wi-Fi link may have more unstable quality and sometimes may be unavailable than LTE link.

However, Wi-Fi can be still utilized for offloading purpose even in mobile environment [8, 9], and its offloading capability may increase in static environment or when

[1] In this chapter, it is assumed that seamless handover of LTE links is ensured. Other cellular links, e.g., 3G/HSDPA, may be applied instead of LTE.

a device associates to an IEEE 802.11ac-compliant Wi-Fi AP that can provide very high throughput. From this perspective, when a DASH client requests video chunks via both Wi-Fi and LTE in parallel in a multi-homed device, Wi-Fi link availability is like a double-edged sword. In other words, although the Wi-Fi link may enhance the video quality at low energy cost and reduce LTE data consumption, it can also increase the possibility of rebuffering events, as we cannot predict the exact bandwidth and availability of Wi-Fi in mobile and dense environments.

Therefore, it is challenging to design a chunk request policy for DASH by judicially utilizing Wi-Fi connectivity in addition to LTE to maximize the merit of utilizing Wi-Fi link while minimizing its side effects. To tackle this challenge, we opportunistically request video chunks over Wi-Fi, thus reducing the side effects of unstable of Wi-Fi links.

1.2 Overview of Existing Approaches

1.2.1 Power modeling of smartphone

The previous efforts on wireless devices' power consumption modeling include the power-throughput curve [10], the airtime and packet rate-based model [11], and the state transition between the high and low power states according to the packet transmission and reception (tx/rx) rates [12–14]. Huang *et al.* [10] model the power used for data transfer via WiFi or cellular interface as a linear function of uplink and downlink throughput. The models in [12, 14] quantize the power consumption of network interfaces into several discrete constant values based on the packet rate, which are too simplified to accurately estimate the power consumption of network interfaces. The authors of [15, 16] group the packets with inter-packet times below a threshold into a burst of packets, and model the power levels of the burst, idle, and sleep modes of a device. Some of these models overlook the power saving operations of the network interfaces in a smartphone [10, 11], or the others model the power of an active state

in a coarse manner, e.g., a single constant value [12–16], thus resulting in significant estimation error in certain scenarios. Most importantly, these power models are limited to single network-activated cases.

Sun *et al.* [17] extend throughput-power linear fitting model considering the packet length and the transport layer protocol, but the model is limited to the WiFi power consumption. In [11], the authors model the power consumption of WiFi devices and investigate per-frame energy consumption, called 'cross-factor.' This work does not consider the power saving operation because it is the power model for the WiFi devices on laptop PCs, which is less sensitive to the power issue than smartphones. Ding *et al.* [18] model the impact of 3G/WiFi signal strength on smartphone energy consumption based on measurement. The authors apply different tx/rx power consumption depending on the RSS of 3G/WiFi to incorporate the effect of the signal strength to the power consumption.

The existing models have two kinds of shortcomings. One is overlooking the power saving operation of smartphones, and the other is modeling the active state power of network interfaces in a coarse manner, thus causing significant estimation error in certain scenarios.

1.2.2 Battery drain rate estimation

The battery information delivered by the battery interface can be used to model the power consumption of smartphones [19,20]. The authors of [19] propose a self-modeling using the discharge current provided by the battery interface. However, battery interfaces in state-of-the-art smartphones rarely provide the discharge current [20], and hence, it is not applicable to most smartphones. V-edge [20] utilizes resistive voltage drop at the internal resistance of the Li-ion battery to realize the fast power consumption modeling of smartphones. In this case, the value of the internal resistance should be known to convert voltage drop into the corresponding discharge current. However, battery resistance varies according to temperature and battery aging.

The authors of [13] utilize SoC to construct smartphone's power consumption models. Instead of SoC, V_{oc} is used to reduce the measurement time [21] using the relationship between V_{oc} and SoC. In addition, the energy loss due to the internal resistance of a Li-ion battery is considered for accurate estimation of remaining battery energy and power modeling of smartphone components [21].

The battery aging of Li-ion batteries for mobile devices is studied in [22]. The authors proposed the method to estimate the degree of aging of an arbitrary Li-ion battery by comparing charging speed of an aged battery to that of a fresh battery.

1.2.3 Optimizing the performance of video streaming

Energy/cellular data quota-aware video streaming: Previous efforts [23,24] control a video segment size to be prefetched in an HTTP-based video streaming service to improve the energy efficiency. *GreenTube* [23] aims to minimize an unnecessary active period of 3G/4G radios by scheduling each video segment downloading. *eSchedule* [24] utilizes crowd-sourced video viewing statistics and power models for energy efficient scheduling of video streaming. *QAVA* [25] manages the tradeoff between cellular data usage and video quality by predicting video client's usage behavior. *QAVA* automatically selects optimal video quality to enable users to keep under their data quota while maximizing video quality. Lee *et al.* [26] and Hu *et al.* [27] consider wastage of energy and cellular data quota incurring when user stops watching video. Both consider the scenario where video streaming is conducted only via LTE interface of a smartphone. *GreenBag* [28] utilizes both LTE and Wi-Fi links to achieve better quality of service (QoS) and energy efficiency. Go *et al.* [29] propose an energy-aware HTTP adaptive video streaming under a cellular data usage constraint. It considers simultaneous usage of LTE and Wi-Fi for DASH video streaming on smartphones. It selects video bitrate and the number of chunks to request via each network in order to minimize a weighted sum of video distortion and energy consumption per video chunk.

Optimal bitrate selection of DASH video: Video client selects a video chunk with an appropriate bitrate according to current network status [30,31] or video player's buffer status [32]. Due to severe fluctuation of link throughput in mobile environment, it is difficult for a link throughput-based bitrate adaptation to practically function, and for this reason, a buffer-based bitrate adaptation has been studied and practically used by real implementations [32, 33].

1.3 Main Contributions

1.3.1 PIMM: Power Modeling of Multiple Network Interface-Activated Smartphone

We propose a packet interval-based power modeling of multiple network interface-activated smartphones (PIMM), which is a generalized and extensible power model for state-of-the-art smartphones. The proposed power model features: 1) accurate estimation of the energy consumption of a smartphone for all kinds of traffic from packet traces, especially with varying packet lengths, intervals, and channel conditions, and 2) extensibility to combine multiple network interfaces' power models, estimating the power consumption when both WiFi and LTE are activated for data communication.

Two key contributions of this chapter can be summarized as follows.

- We propose a 'packet interval-based power modeling' for smartphones which has the valuable features as described above.
- Using real-world applications and emulations, we validate PIMM shows quite good accuracy for both single and multiple network-activated scenarios.

With these contributions, the PIMM can be used for the development and evaluation of energy-efficient smartphone operations. For example, it can help evaluate the performance of specific multiple network interface activation algorithms in the

simulation tools so that the algorithm developers do not need to put much effort into measuring the device's power to evaluate the performance.

1.3.2 BattTracker: Real-Time Battery Drain Rate Estimation

To enable instantaneous battery drain monitoring taking temperature and battery aging into consideration, we propose *BattTracker*, a novel battery drain rate/lifetime estimation algorithm for mobile devices in real-time using battery characteristics. Specifically, we design *BattTracker* to achieve the following goals: (a) it accurately provides instantaneous battery drain rate, (b) it estimates battery drain rate considering both temperature and battery aging, and (c) most importantly, we develop a solution which does not require any training effort to figure out the detailed impact of temperature and battery aging on battery characteristics.

With instantaneous battery drain rate estimated by *BattTracker*, Users can manage their battery lifetime by themselves and mobile application developers can embed energy-awareness into their applications such as video streaming, web surfing, and file transferring to optimize energy efficiency or to guarantee the battery lifetime.

Contributions of *BattTracker* are summarized as follows:

- Through measurement, we model the effects of temperature and aging on capacity and internal resistance of Li-ion batteries used in smartphones.
- We propose the effective resistance concept, which helps estimate battery drain rate without knowing the detailed impact of temperature and aging on battery characteristics.
- We propose *BattTracker*, which estimates battery drain rate during run time for any temperature and any degree of battery aging with fine-grained time resolution.
- The performance of *BattTracker* is extensively evaluated with smartphones and differently aged batteries for various mobile applications with varying temperature, and confirm that *BattTracker* estimates battery drain rate with high accuracy.

1.3.3 REQUEST: Performance Optimization of DASH Video Streaming

To provide an optimal video performance while satisfying resource usage constraints for battery energy and LTE data quota, we propose REQUEST, a bit*R*ate, *E*nergy, LTE data *Q*uota, and b*U*ff*E*r-aware video *ST*reaming, for DASH video over multi-homed smartphones. By utilizing Wi-Fi link as a supplementary link, REQUEST realizes a seamless DASH video streaming even in the environments where Wi-Fi link performance is not guaranteed. Also, REQUEST optimizes time-average video quality while satisfying time-average resource constraints by adopting Lypunov optimization framework, and it is easily implemented in commercial smartphones as an application. We claim the following major contributions:

- We propose a chunk request policy that achieves seamless playback of video using both Wi-Fi and LTE simultaneously even in situations where Wi-Fi is unstable.
- We formulate a Lyapunov optimization framework-based stochastic optimization problem to maximize time-average video quality under time-average energy and LTE data usage constraints and minimize rebuffering.
- We design REQUEST, an online video chunk request algorithm by using both LTE and Wi-Fi links, which provides a near-optimal solution of the Lyapunov optimization problem.
- We implement REQUEST by modifying ExoPlayer, Google's open-source DASH player for Android [34], and validate its performance in real-world scenarios.

1.4 Organization of the Dissertation

The rest of the dissertation is organized as follows.

Chapter 2 presents PIMM, a packet interval-based power modeling of multiple network interface-activated smartphones. Power saving operations of Wi-Fi and LTE are

introduced and a modeling methodology and packet interval-based power model are provided. Practical issues, i.e., channel quality and packet length are considered. The modeling accuracy of PIMM are validated through extensive evaluations in various scenarios.

In Chapter 3, we present *BattTracker*, a real-time battery drain rate estimation algorithm for smartphones. First, we provides the background for battery interface and Li-ion battery. Next, we describe Li-ion battery characteristics and introduce the concept of effective resistance. The design and key algorithms of *BattTracker* are presented and the performance of *BattTracker* is evaluated in various scenarios with differently aged batteries.

Chapter 4 presents REQUEST, an online video chunk request policy that achieves seamless playback of video using both Wi-Fi and LTE simultaneously. We first discuss is issues arising when utilizing both LTE and Wi-Fi for DASH video streaming and related work. Our proposed video chunk request policy is presented and we provide REQUEST algorithm based on Lyapunov optimization problem. We comparatively evaluate the performance of REQUEST via measurements with modified Google's open-source DASH client, ExoPlayer.

Finally, Chapter 5 concludes the dissertation with the summary of contributions and discussion on the future work.

Chapter 2

PIMM: Packet Interval-Based Power Modeling of Multiple Network Interface-Activated Smartphones

2.1 Introduction

Today's smartphones can activate and use multiple network interfaces, e.g., Wi-Fi and LTE, simultaneously to achieve higher throughput or to maintain seamless connection to a server. The *download booster* of Samsung Galaxy S5[1] utilizes Wi-Fi and LTE networks simultaneously to boost up the file download speed. Apple's *iOS 7* supports MPTCP (Multi-Path TCP) to enhance the reliability of *Siri* services by using Wi-Fi as the primary TCP connection and cellular data as a backup connection. In addition, mobile hotspot service (a Wi-Fi-based tethering service) and *Airplug*[2] (a video streaming application platform using both LTE and Wi-Fi to provide stable video quality) are also the real-world applications which simultaneously utilize both LTE and Wi-Fi at the same time. In this case, the energy consumption of a smartphone varies depending on the number of activated network interfaces and their usage.

Meanwhile, measuring the device power consumption using a power meter to evaluate the energy efficiency of a certain algorithm is a fairly burdensome job, thus mo-

[1] http://www.samsung.com/global/microsite/galaxys5/features.html.

[2] http://www.airplug.com/solution.php.

tivating researchers to develop power consumption models. The previous efforts on wireless devices' power consumption modeling include the power-throughput curve [10], the airtime and packet rate-based model [11], and the state transition between the high and low power states according to the packet transmission and reception (tx/rx) rates [12–14]. The authors of [15, 16] group the packets with inter-packet times below a threshold into a burst of packets, and model the power levels of the burst, idle, and sleep modes of a device. Some of these models overlook the power saving operations of the network interfaces in a smartphone [10, 11], or the others model the power of an active state in a coarse manner, e.g., a single constant value [12–16], thus resulting in significant estimation error in certain scenarios. Most importantly, these power models are limited to single network-activated cases.

Being motivated by this, we propose a packet interval-based power modeling of multiple network interface-activated smartphones (PIMM), which is a generalized and extensible power model for state-of-the-art smartphones. The proposed power model features: 1) accurate estimation of the energy consumption of a smartphone for all kinds of traffic from packet traces, especially with varying packet lengths, intervals, and channel conditions, and 2) extensibility to combine multiple network interfaces' power models, estimating the power consumption when both Wi-Fi and LTE are activated for data communication.

Two key contributions of this chapter can be summarized as follows. First, we propose a 'packet interval-based power modeling' for smartphones which has the valuable features as described above. Second, using real-world applications and emulations, we validate PIMM shows quite good accuracy for both single and multiple network-activated scenarios. With these contributions, the PIMM can be used for the development and evaluation of energy-efficient smartphone operations. For example, it can help evaluate the performance of specific multiple network interface activation algorithms in the simulation tools so that the algorithm developers do not need to put much effort into measuring the device's power to evaluate the performance.

The rest of this chapter is organized as follows. Section 2.2 presents the background of this work. Section 2.3 provides the proposed power modeling, PIMM, and practical issues are considered in Section 2.4. Section 2.5 comparatively evaluate the performance of PIMM in various scenarios. Finally, Section 2.6 concludes the chapter.

2.2 Background

2.2.1 Power saving operations of Wi-Fi and LTE

Wi-Fi network interface: IEEE 802.11 standard defines two power management modes, namely, **active mode (AM)** and **power save mode (PSM)**. The Wi-Fi interface in **AM** always runs in the **awake state** in which it can transmit and receives the packets, while the Wi-Fi interface in **PSM** toggles between the **awake state** and **doze state**, in which the interface is basically turned off. The AP informs the presence of the buffered packets via the traffic indication map (TIM) element in the beacons. The Wi-Fi interface in **PSM** periodically wakes up, i.e., switching from **doze state** to **awake state**, for every delivery TIM (DTIM) period to receive beacons, and checks the TIM element in the beacon. If there exist buffered packets destined to the device, the Wi-Fi interface sends a null data packet (NDP) in order to switch from **PSM** to **AM**, and then receives the buffered packets. The Wi-Fi interface sets the inactivity timer which has a specific timeout value, called *PSM timeout*, and it resets the timer for every packet reception. When the inactivity timer expires, the Wi-Fi interface switches back to **PSM** to save energy by sending an NDP to the AP.

LTE network interface: LTE has two radio resource control (RRC) states, namely, **RRC_connected** and **RRC_Idle**. Once an eNodeB (i.e., LTE base station) allocates resources to a user equipment (UE), the UE operates at **RRC_connected** state consuming high power. At **RRC_connected** state, the UE first stays in **continuous reception** mode and keeps monitoring physical downlink control channel (PDCCH). The UE starts **discontinuous reception (DRX) operation** and enters **short DRX** mode

when *DRX inactivity timer* expires. The UE remains in *short DRX* until *short DRX cycle timer* expires and goes to **long DRX** when it expires. If the UE, which operates in **long DRX**, does not receive any packet until *RRC inactivity timer* expires, it enters **RRC_idle** state by releasing the allocated resources, and saves the energy consumption. When the UE operates in either **short DRX** or **long DRX**, it goes back to **continuous reception** if it receives/transmits a packet. During **RRC_idle** state, the UE sleeps for most of time and periodically wakes up to receive paging messages from eNodeB. Accordingly, it is awake for a few milliseconds every *RRC_idle DRX cycle* period, which is for example 1.28 seconds.

2.2.2 Related work

Research on power modeling of network interfaces: The models in [12, 14] quantize the power consumption of network interfaces into several discrete constant values based on the packet rate, which are too simplified to accurately estimate the power consumption of network interfaces. Huang *et al.* [10] model the power used for data transfer via Wi-Fi or cellular interface as a linear function of uplink and downlink throughput. Sun *et al.* [17] extend throughput-power linear fitting model considering the packet length and the transport layer protocol, but the model is limited to the Wi-Fi power consumption. In [11], the authors model the power consumption of Wi-Fi devices and investigate per-frame energy consumption, called 'cross-factor.' This work does not consider the power saving operation because it is the power model for the Wi-Fi devices on laptop PCs, which is less sensitive to the power issue than smartphones. The authors of [15, 16] group packets of which the inter-packet time is less than a certain threshold into a burst of packets, and model the power levels of the burst (active), idle, and sleep modes of a device. Ding *et al.* [18] model the impact of 3G/Wi-Fi signal strength on smartphone energy consumption based on measurement. The authors apply different tx/rx power consumption depending on the RSS of 3G/Wi-Fi to incorporate the effect of the signal strength to the power consumption. The existing models

have two kinds of shortcomings. One is overlooking the power saving operation of smartphones, and the other is modeling the active state power of network interfaces in a coarse manner, thus causing significant estimation error in certain scenarios.

Multi-homed device and MPTCP: ESPA [35] utilizes the multiple network interfaces of smartphones, i.e., Wi-Fi and LTE, to enhance the user experience for FTP services. It considers the energy consumption and cellular data quota in addition to the file download completion time. In particular, ESPA experimentally characterizes the power consumption of the network interfaces and CPU to obtain the best network selection in terms of energy efficiency. MPMTP [36] exploits path diversity over multiple network interfaces to provide a seamless video streaming service by using Raptor coded multiple network interfaces simultaneously and Raptor coded UDP transmissions. Peng *et al.* [37] design an MPTCP algorithm by jointly considering the performance and energy consumption. They conducted simulation with the MPTCP power model given by $\sum_{r \in S} P_r(x_r)$, where $S \in \{\text{3G, 4G, Wi-Fi}\}$. $P_r(x_r) = a_r x_r + b_r$, where x_r is the throughput over interface r, and a_r and b_r are the coefficients of the throughput-based power model in [10]. Lim *et al.* [38] develop an energy consumption model for MPTCP based on the experimental measurement using a smartphone. Through experimental measurement, the authors observed that the MPTCP energy consumption is less than the sum of energy consumed by each network interface, i.e., Wi-Fi and LTE. The authors introduce the constant coefficient γ to make the estimated power by their model fit with the measurement. As evaluated in Section 2.5.3 in comparison with our model, the models in [37, 38] do not properly capture the characteristics of multiple network-activated smartphones.

2.3 Power Consumption Modeling

In this section, we propose a power consumption model for the smartphones, capable of activating/using multiple network interfaces simultaneously. The best way to

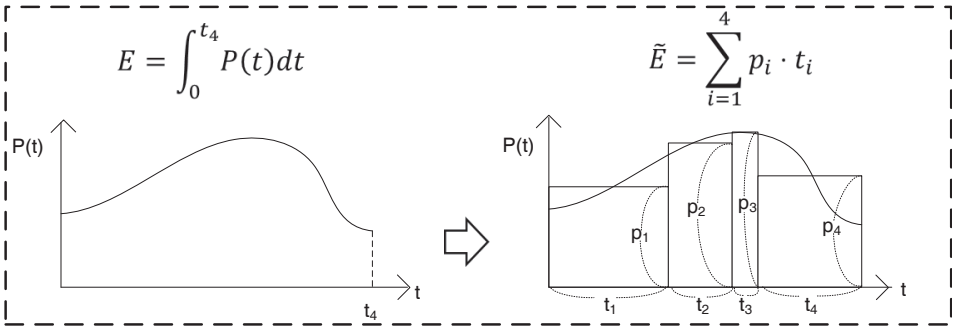
estimate energy is integrating the instantaneous power consumption over time. However, it is impossible to obtain instantaneous power consumption without any external power/current measuring equipment with high granularity and precision. For this reason, the existing approaches have tried to approximate the energy consumption by converting certain representative factors, e.g., average-sense throughput and packet rate, network states (active/idle) given by system calls, into the modeled power values. These models have their own drawbacks and are not easily generalized to cover all kinds of traffic and network status. Moreover, the scenarios utilizing multiple network interfaces simultaneously have not been accurately modeled. Being motivated by this, we propose a generalized and extensible power model for smartphones, thus making it possible to incorporate as many situations as possible, including the number of simultaneously activated network interfaces, packet size, and channel conditions.

2.3.1 Modeling Methodology

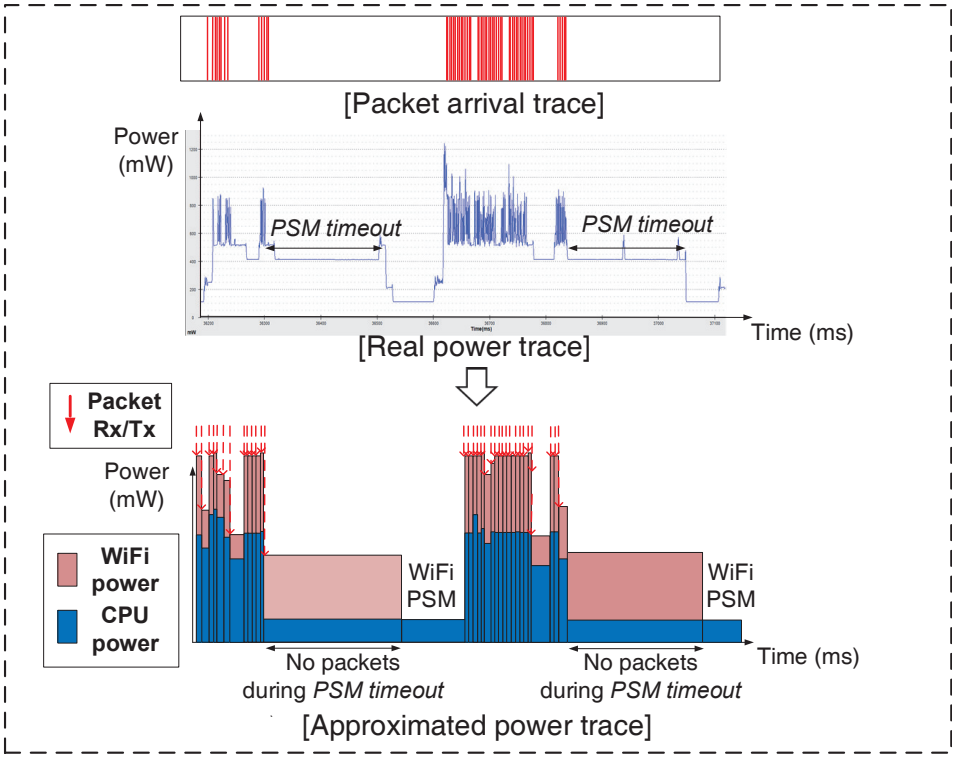
Packet interval-based energy approximation

Our energy estimation methodology for data communication is based on the integration of the approximated power consumption over time. To approximately obtain the energy consumption, we divide the total time in consideration into a countable number of time partitions. Fig. 2.1b represents an example that imitates a smartphone's real power trace from a given packet trace and generates the approximated power trace. The power consumption is estimated with the rules as follows: 1) a packet interval becomes a partition size, e.g., $t_i - t_{i-1}$ in Fig. 2.1a, and 2) the average power consumption for the partition i is represented by a certain value that is equivalent to the average power for the partition, e.g., p_i in Fig. 2.1a.

Dividing the total time into the packet intervals has two kinds of merit. First, the representative power value p_i for interval t_i, called *per-packet* power consumption, can be effectively obtained by measuring the power consumption of the smartphone that sends/receives packets periodically generated with interval t_i at the modeling stage



(a) The example of approximating the area of the region underneath the curve $P(t)$.



(b) The energy consumption approximation of a smartphone which communicates using a Wi-Fi interface.

Figure 2.1: The area approximation methodology (a) and the power estimation approach from a given packet trace (b).

without knowing the exact tx/rx artime of the packet. Second, the power saving operations of the network interfaces are highly related to the packet interval, and hence, the power saving operations can be easily considered by setting the packet interval as one of the modeling variables even if no system call-based finite state machine [12] is managed. With these advantages, our modeling approach accurately estimates the power consumption caused by the network operations irrespective of traffic pattern, and even provides the insight of the energy efficiency with respect to the traffic pattern, e.g., burst traffic results in higher energy efficiency [10]. In the following, the definition and characteristics of *per-packet* power consumption of the network interfaces are explained.

Per-packet power of network interface

The *per-packet* power consumption, P_{pp}, is defined as the average power consumption incurred by a packet until the corresponding network interface goes to the low power mode (**LPM**), i.e., **PSM** and **RRC_idle** for Wi-Fi and LTE, respectively. Whenever a packet reception is completed, the network interface waits for a while in the active state, and enters the LPM when the *LPM timer* expires, where the *LPM timer* has a predefined timeout value, namely *LPM timeout*. Then, the tail time is defined as the time duration from the end of packet reception to the next packet arrival time, e.g, T_{tail}^{i} and T_{tail}^{i+1} in Fig. 2.2, or to the time when the *LPM timer* expires, e.g., T_{tail}^{i+2}. The **LPM**, *LPM timeout*, and power consumption for the tail time of the Wi-Fi/LTE interface are summarized in Table. 2.1.

P_{pp} is the function of the packet interval, and is defined only for the packet intervals less than the *LPM timeout* of a network interface. Fig. 2.2 represents a simple example of P_{pp} for sending the packets via a network interface, i.e., $P_{pp} = \frac{P_{tx} \cdot T_{tx} + P_{tail} \cdot T_{tail}}{T_{tx} + T_{tail}}$, where P_{tx}/P_{tail} and T_{tx}/T_{tail} are the tx/tail power consumption and time durations, respectively. If the packet interval $T_{interval}$ is larger than $T_{tx} + T_{to}$, P_s is applied for T_s ($= T_{interval} - T_{tx} - T_{to}$). The energy consumption for $T_{interval}$ is calculated by

Table 2.1: Terminologies of power saving operations.

	LPM	*LPM timeout* (T_{to})	Tail power (P_{tail})
Wi-Fi	**PSM**	*PSM timeout*	channel sensing power
LTE	**RRC_idle**	*RRC inactivity timer timeout*	**long DRX** state power

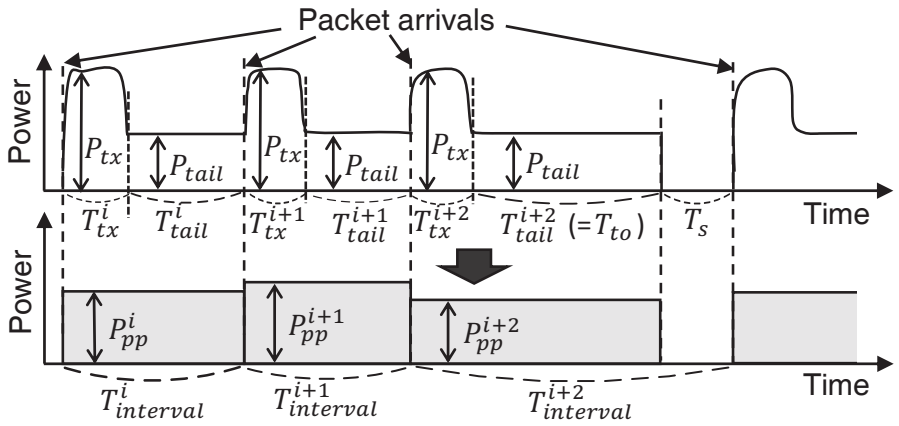


Figure 2.2: Per-packet power consumption model.

$P_{pp} \cdot (\min(T_{interval}, T_{to} + T_{tx})) + P_s \cdot T_s \cdot u(T_{interval} - T_{tx} - T_{to})$, where $u(\cdot)$ is the unit step function. P_s in Fig. 2.2 is zero and normally $T_{to} + T_{tx} \simeq T_{to}$ since $T_{to} \gg T_{tx}$.

P_{pp} is mainly determined by the ratio of the tx/rx airtime to the tail time of the network interface. Without the tail time, e.g., sending an aggregate MAC protocol data unit (A-MPDU) of IEEE 802.11n, i.e., $T_{tail} = 0$ between the MAC frames, P_{pp} is equal to P_{tx}. In the case of a packet sent via the LTE interface, P_{pp} converges to the LTE **long DRX power**, called LTE tail power, if the packet interval is close to the LTE *RRC inactivity timer*, i.e., $T_{to} \gg T_{tx}$. In short, P_{pp} is a decreasing function of the packet interval but lower-bounded by P_{tail}. We highlight that the power model is extended to the cases that consider the impacts of the packet length and channel status, i.e., the channel quality and congestion. Fig. 2.3 summarizes how these factors affect

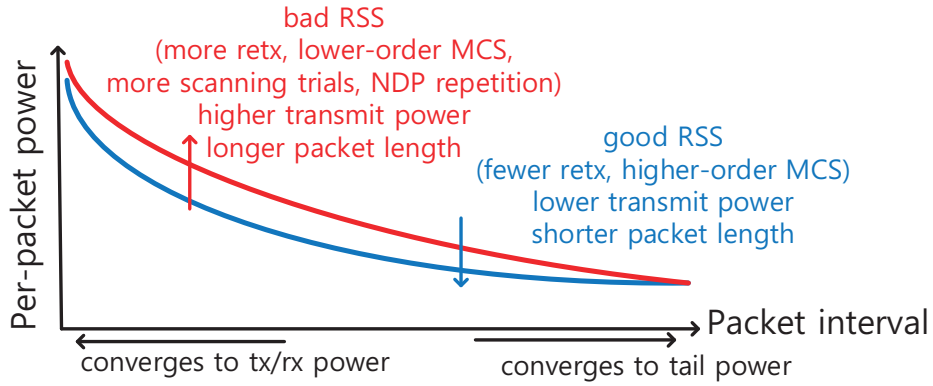


Figure 2.3: Summary of per-packet power characteristics.

P_{pp}. Congested channel, smaller available TCP window (due to long round trip time (RTT)), and lower packet generation rate lead to the increase of the packet interval, thus resulting in lower P_{pp}. Otherwise, higher packet generation rate, less contention, and shorter RTT make the packet interval shorter, and hence, P_{pp} goes to high. Meanwhile, even for the same packet interval, bad quality channel, i.e., low received signal strength (RSS), and long packet length incur higher P_{pp} compared to the good quality channel and small packet length due to increased tx/rx time portion and retransmission (retx) rate. The details of the P_{pp} modeling are presented in Section 2.3.2.

Multiple network-activated smartphone

Even though multiple network interfaces of a smartphone are activated and operated independently, all the packets are processed at a common CPU of the smartphone after received or before transmitted. Accordingly, the power of the multiple network interface-activated smartphone is modeled as the sum of the power of each network interface and the CPU packet processing power for the aggregated packet rate. Fig. 2.4 shows the packet flow between the applications and the network interfaces. For the case of the packet reception, the set of data packets received by the network interfaces for time T is represented as $\mathbb{P}_{network} = \mathbb{P}_{Wi-Fi} \cup \mathbb{P}_{lte}$, where $\mathbb{P}_{Wi-Fi}$ and $\mathbb{P}_{lte}$ are the

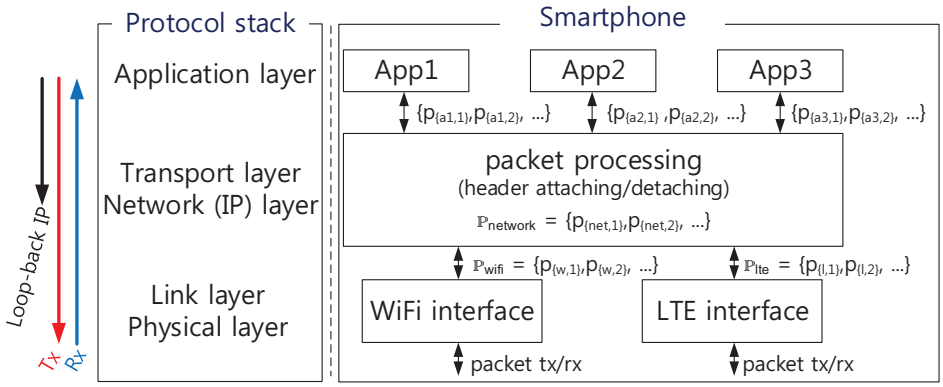


Figure 2.4: Packet flow procedure at a multiple network interface-activated smartphone.

set of the packets received via the Wi-Fi and LTE interfaces, respectively. The packets in $\mathbb{P}_{network}$ pass through the protocol stacks, incurring the packet processing power consumption at the CPU. Finally, the payloads are delivered to the corresponding applications. The packet transmission procedure is similar to the packet reception. On the other hand, if the destination address of a packet is set to the loop-back IP address, the packet is processed only upon the IP-layer and not delivered to the network interface, incurring only the packet processing power consumption.

With this overview, the power consumption is decomposed into two main parts, namely, 1) the power P_{Wi-Fi}/P_{lte} of the network interfaces when sending/receiving packets with their antennas and 2) the power P_{proc} of the CPU when processing the packets. Consequently, in terms of data communication, the power consumption of multiple network interface-activated smartphone is obtained by synthesizing each individual power model. We provide the details of the power models for CPU packet processing, Wi-Fi, LTE, and their synthesis in the following section.

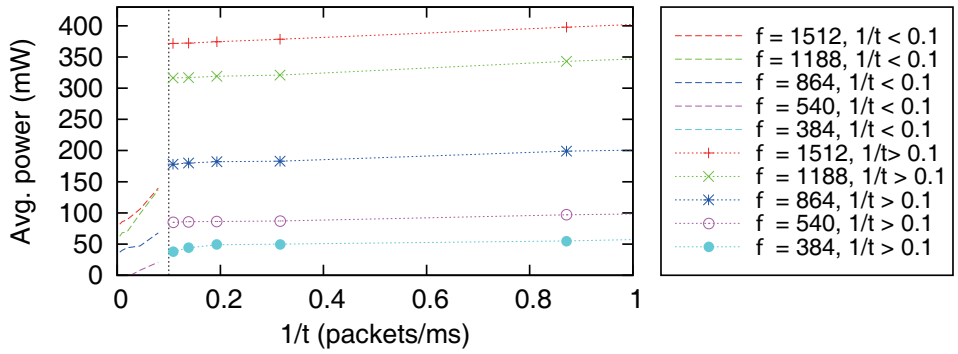


Figure 2.5: CPU packet processing power with respect to $1/t$ (packets/ms) according to the CPU clock frequency f (MHz).

2.3.2 Packet interval-based power modeling

In this section, we provide the *per-packet* power modeling with the extensive measurement. We use Samsung Galaxy S2 (SHV-E120) and Galaxy S4 (SHV-E330) smartphones with Android OS 4.1.2 and Monsoon power monitor for modeling the power consumption. The *tcpdump* is used to capture the packet traces of the smartphones. Moreover, for Wi-Fi, the packets captured by *Airpcap* are also used to obtain the detailed information of the packets over the air.

Per-packet power model of CPU processing

We first analyze how much the CPU power is consumed for packet processing. The CPU packet processing power, P_{proc}, is related to the packet rate, γ, i.e., the reciprocal of packet interval t^{-1}, regardless of the packet length [11]. P_{proc} is affected by the CPU clock frequency, which is adaptively changed according to the CPU usage by DVFS (dynamic voltage and frequency scaling), one of the CPU power management techniques.

To measure the CPU power consumption for packet processing, we implement a simple socket program on the smartphone and sends the packets by setting the destination IP address to the loop-back IP address ('127.0.0.1'). Fig. 2.5 is the power

Table 2.2: Coefficients of a linear model for the CPU packet processing power (unit: power (mW), t (ms)).

CPU clock frequency (f_c)	$P_{proc}(t^{-1}) = a \cdot t^{-1} + b$				u^*_{cpu}	$u_p(t^{-1}) = a \cdot t^{-1} + b$		$P_{cpu\_o}(u_o) = a \cdot u_o + b$		P_{base}
	$(u_{cpu} < u^*_{cpu})$		$(u_{cpu} > u^*_{cpu})$							
	a	b	a	b		a	b	a	b	
384 MHz	398.1	32.7	16.2	106.9	16.8	4.3	16.4	1.5	106.6	105
540 MHz	361.8	34.9	12.1	127.1	13.4	4.3	12.9	2.4	127.1	107
864 MHz	385.1	55.1	18.0	178.2	10.5	7.9	9.8	4.0	198.1	130
1188 MHz	100.6	12.5	31.3	269.9	7.1	5.1	6.6	7.3	259.9	150
1512 MHz	762.7	11.5	33.4	303.0	6.3	9.7	5.3	10.2	292.8	170

consumption when the smartphone sends packets with the periodic packet interval t (ms) using the loop-back IP for various CPU clock frequencies. The higher CPU clock frequency, the higher power consumption even for the same number of packets. The packet processing power is well modeled as a discontinuous piecewise linear function of the inverse of packet interval t^{-1} with one step at $t^{-1} = 0.1$ (packets/ms). When the packet rate is low, i.e., the packet interval is longer than the CPU tail time, the CPU can switch to the low power state until the next packet arrives. Otherwise, the CPU continuously processes the packets in the high power state without switching to the low power state. Reflecting this observation, we summarize the coefficients of the discontinuous piecewise linear model for the CPU packet processing power, $P_{proc}(t^{-1}) = a \cdot t^{-1} + b$, in Table 2.2.

The modeling described above is valid only when any other applications using the CPU are not activated, e.g., the case that a smartphone transmits/receives packets in a background job. For real applications, the other processes may use the CPU at the same time. Therefore, the CPU remains at the high power state when the total CPU usage u_{cpu} exceeds the CPU usage corresponding to processing packets with 0.1 (packets/ms) packet rate, which is the discontinuous point of the function in Fig 2.5. To model the CPU power consumption for the general cases, we decompose the total CPU usage u_{cpu} $(= u_p + u_o)$ into two parts, i.e., u_p and u_o, which are the CPU usage by the processing packets and the other operations, respectively. Considering the base CPU power consumption, the CPU power consumption model is constructed as follows:

$$P_{cpu} = P_{base} + P_{proc} + P_{cpu\_o}(u_{cpu} - u_p), \tag{2.1}$$

where P_{cpu}, P_{base}, P_{proc} and P_{cpu_o} are the total CPU power, CPU base power, packet processing power, and the other operations' CPU power, respectively. u_p and P_{cpu_o} are modeled by linear functions of the packet rate and CPU usage, respectively. u^*_{cpu} is the CPU usage threshold to determine whether the CPU remains at the high power state or not, and it is equal to $u_p(0.1)$. The coefficients of the models are obtained by running a training application with varying CPU clock frequencies and summarized in

Table 2.2.

Per-packet power model of network interfaces

Now, we propose a new power model for the Wi-Fi/LTE network interface, i.e., *packet interval-based power function approximation model*. First of all, we experimentally investigate the relationship between the packet interval and average power consumption. We implement the socket programs on the smartphone and a laptop to configure the packet interval. The measured power consumption of the network interface with the packet interval t is obtained by subtracting the packet processing power, $P_{proc}(t^{-1})$ in Table 2.2 from the smartphone's overall power consumption measured by the power monitor. Through intensive measurement, we validate that the power consumption of the network interface P_{net} for the packet interval t, where t is smaller than the *LPM timeout* T_{to}, is well fitted to the *power function* ($f : x \mapsto \alpha \cdot x^{\beta}$, where $\alpha, \beta \in \mathbb{R}$) with respect to t as follows:

$$P_{net}(t) = \max(\alpha \cdot t^{\beta}, P_{tail}), \;\; t < T_{to}. \tag{2.2}$$

Fig. 2.6 shows the measured power when the smartphone sends 1400-byte UDP packets over 5 GHz Wi-Fi with varying packet intervals, and estimated power by (2.2) with $\alpha = 723.44$ and $\beta = -0.373$. The estimated power and measured power quite well match. For the Wi-Fi interface, *PSM timeout* is 210 ms and the channel sensing power is 300 mW for 5 GHz band and 180 mW for 2.4 GHz band from our measurement by sending the packets with increasing intervals. We find that **long DRX** power is 880 mW and *RRC inactivity timer timeout* is 10.8 s using the same approach as the Wi-Fi interface case. (2.2) is lower bounded by P_{tail}, i.e., the channel sensing power for Wi-Fi and **long DRX** state power for LTE, as explained in Section 2.3.1. To obtain the coefficients of the power model (2.2) empirically, we make the smartphone send and receive the packets via Wi-Fi and LTE with various periodic packet intervals while measuring the power consumption. In Table 2.3, we summarize the coefficients of the *power function* approximation for the Wi-Fi and LTE *per-packet* power models.

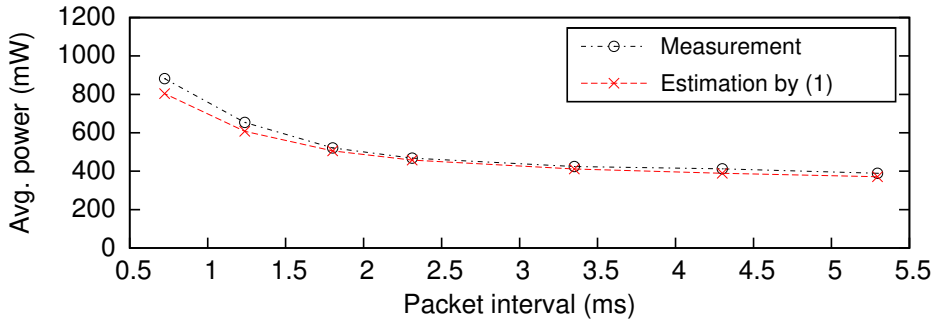


Figure 2.6: Measured and estimated power consumption of Wi-Fi interface for 1400-byte UDP packet transmission over 5 GHz band.

Table 2.3: Coefficients of power function approximation for the 1400-byte packet tx/rx power consumption of network interfaces.

(for t ms)	tx			rx		
$\alpha \cdot t^{\beta}$	α	β	R^2	α	β	R^2
Wi-Fi 2.4	782.02	-0.408	0.97	446.56	-0.32	0.96
Wi-Fi 5	723.44	-0.373	0.96	527.65	-0.24	0.95
LTE	1853.5	-0.189	0.96	1265.9	-0.125	0.99

The coefficient of determination, R^2, of the approximation is at least 0.95 ($R^2 = 1$ means the approximation is perfectly fitted), and hence, we confirm the feasibility of the *power function* approximated *per-packet* power model.

PIMM: Proposed power model

As mentioned in Section 2.3.1, the power consumption of a multiple network interface-activated smartphone is obtained by combining the individual power models in Sections 2.3.2 and 2.3.2 using the modeling methodology explained in Section 2.3.1. When a data communication happens over time T, the average power consumption P for T is estimated as follows:

$$P = \frac{1}{T}\left(\sum_{i=1}^{n_w+n_l} P_{proc}(\frac{1}{t_i^c})t_i^c + \sum_{j=1}^{n_w} P_{Wi-Fi}(t_j^w)t_j^w + \sum_{k=1}^{n_l} P_{lte}(t_k^l)t_k^l\right), \tag{2.3}$$

where n_w and n_l are the total numbers of data packets received and transmitted via the Wi-Fi and LTE interfaces, respectively. t^w and t^l are the packet intervals of the packets passing through the Wi-Fi and LTE interfaces, respectively, and t^c are the packet intervals of all the packets processed by the CPU as shown in Fig 2.7. P_{proc}, P_{Wi-Fi}, and P_{lte} are the *per-packet* power consumption of the CPU, Wi-Fi and LTE interfaces, respectively. (2.3) holds when t^w and t^l are less than the corresponding *LPM timeout* since P_{Wi-Fi} and P_{lte} are only defined by the packet interval less than *LPM timeout*. For example, if t^w is larger than *PSM timeout* t_{psm}, then $P_{Wi-Fi}(t_j^w)t_j^w$ is substituted by $P_{Wi-Fi}(t_{psm})t_{psm}+p_s t_s$ in order to reflect the energy reduction by **PSM**, where P_s is the Wi-Fi sleep state power[3] and $t_s = t_j^w - t_{psm}$. We refer to (2.3) as PIMM, meaning Packet Interval-based power Modeling of Multiple network interface-activated smartphones. PIMM synthesizes all the power models previously discussed, i.e., the packet processing power and Wi-Fi and LTE interfaces power model. We validate PIMM by

[3]We measure that the Wi-Fi/LTE consume 0.8 mJ/7.2 mJ during beacon/downlink transmission reception, and hence, $P_s = \frac{0.8}{0.3072} = 2.6$ mW for Wi-Fi and $P_s = \frac{7.2}{1.28} = 5.6$ mW for LTE when the DTIM period is 307.2 ms and the RRC_idle DRX cycle period is 1.28 s, respectively.

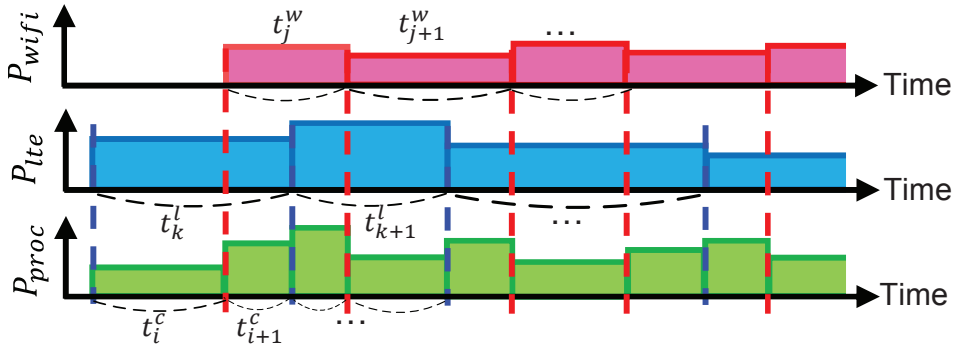
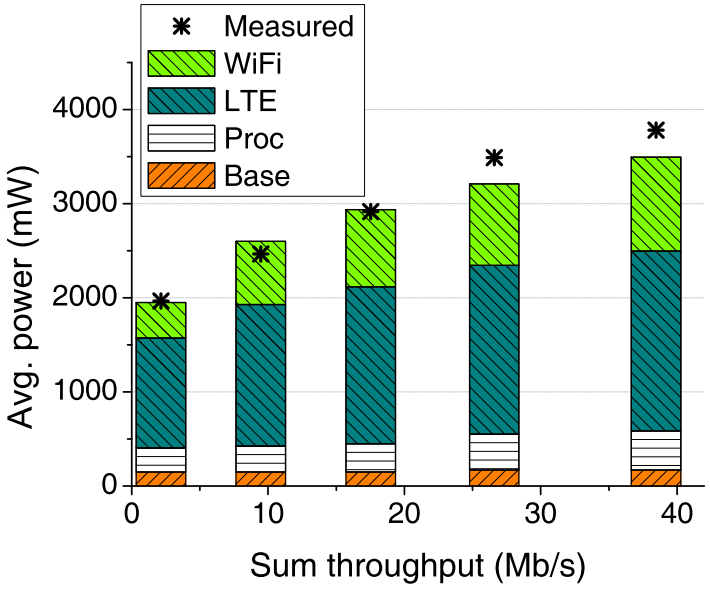


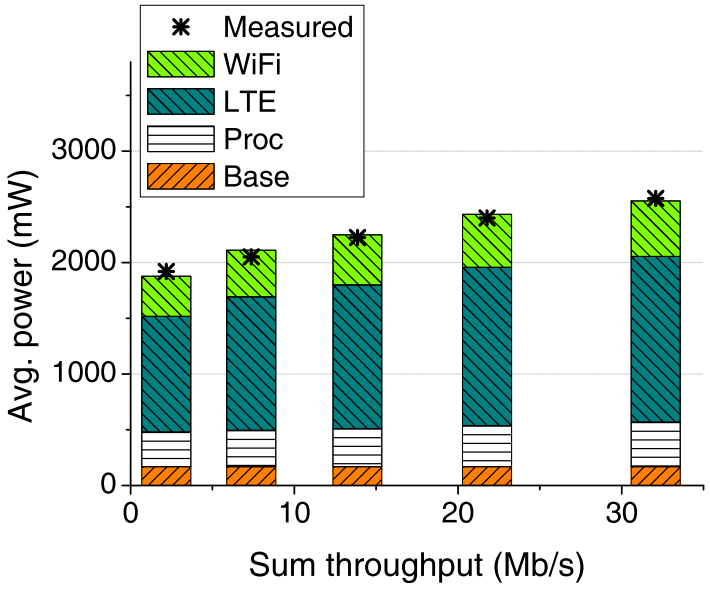
Figure 2.7: Packet interval-based power consumption anatomy for data communication using the multiple network interfaces.

simultaneously activating TCP connections via Wi-Fi and LTE. **Simultaneous activation of Wi-Fi and LTE:** The default connectivity service of Android does not permit to activate both the Wi-Fi and LTE data connections simultaneously, i.e., when the smartphone is associated with an AP, the LTE data connection is disabled and cannot be activated unless the Wi-Fi link is disconnected. However, Android provides the API, called *ConnectivityManager*, with which a user can control the connections at the application layer.

With this API, we can activate both Wi-Fi and LTE and selectively route specific packets via a desired network interface as follows. First, we make the smartphone connect to a Wi-Fi AP, and then activate the LTE data connection by using *startUsingNetworkFeature* function with *TYPE_MOBILE* as the network type variable and *enableHIPRI* as the network feature variable. Second, we generate some traffic to the specified destination delivered via the specified network interface, i.e., the LTE interface, by using *requestRouteToHost* function with *TYPE_MOBILE_HIPRI* as the network type variable. *HIPRI* APN (access point name) type uses the mobile data connection to route specific traffic to mobile network until *HIPRI* timer expires, even if a Wi-Fi connection is alive. Only the process that activates the mobile data connection with *HIPRI* type will have access to the mobile DNS server, and only the traffic



(a) TCP tx.



(b) TCP rx.

Figure 2.8: Model validation of PIMM for various source throughput.

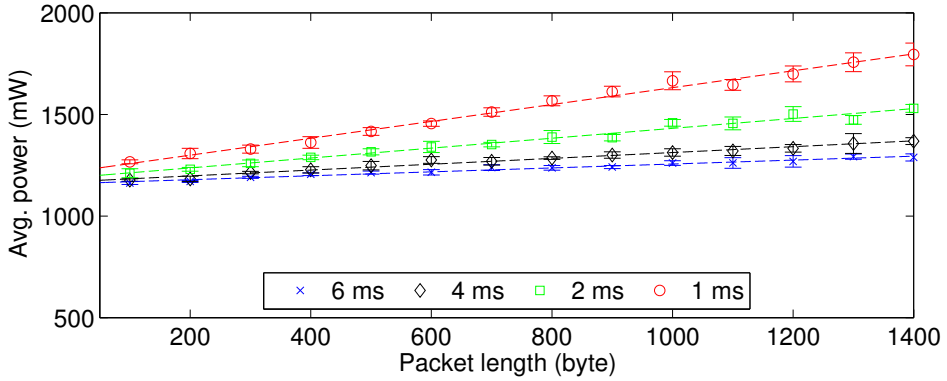


Figure 2.9: Impact of packet length to tx power consumption of LTE interface under periodic packet intervals, 6, 4, 2, and 1 ms. The points and bars represent the average and standard deviation of 10 iterations, respectively.

having IP addresses requested via *requestRouteToHost* will be routed through the mobile network interface. Third, we set the DNS and add the default gateway for the LTE in addition to the Wi-Fi DNS and Wi-Fi gateway by using *setprop net.dns1* and *route add default gw* commands, respectively. After setting up the parallel TCP connections via Wi-Fi and LTE, we make the smartphone send/receive packets via both of the interfaces at the same time.

PIMM model validation: We validate the accuracy of PIMM with various packet generation rates at the application-layer (app-layer) at the server side. Both Wi-Fi and LTE TCP connections are activated and communicate with two local servers using each link. The packet length is 1400 bytes and the packet generation rates for two links are the same. Fig. 2.8 shows the average estimated power consumption by (2.3) for various packet generation rates. The x-axis represents the sum tx/rx throughput of the TCP connections via Wi-Fi and LTE. The average estimated power consumption is composed of "Wi-Fi" P_{Wi-Fi}, "LTE" P_{lte}, "Proc" P_{proc}. We add the CPU base ("Base") power to the estimated power in order to fairly compare with the measured power. PIMM shows quite accurate average power estimation with the estimation error of 4.5$\pm$3.1% for TCP tx and 1.7$\pm$0.8% for TCP rx. We further evaluate the per-

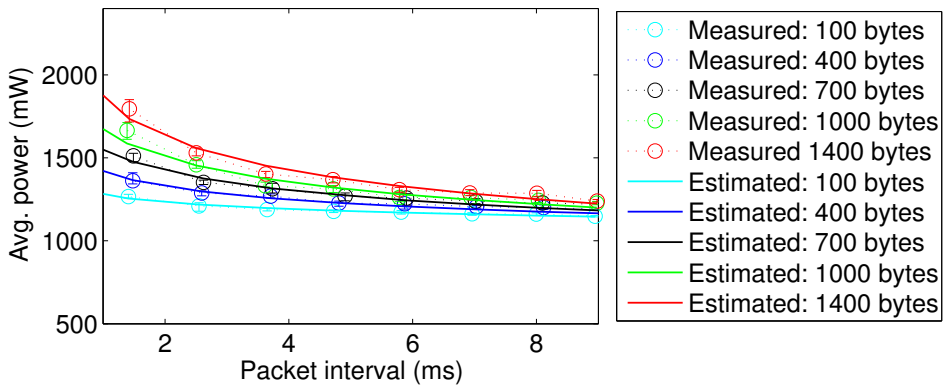


Figure 2.10: Measured and estimated tx power consumption of LTE interface with respect to the packet interval for $\{100, 400, 700, 1000, 1400\}$-byte packets.

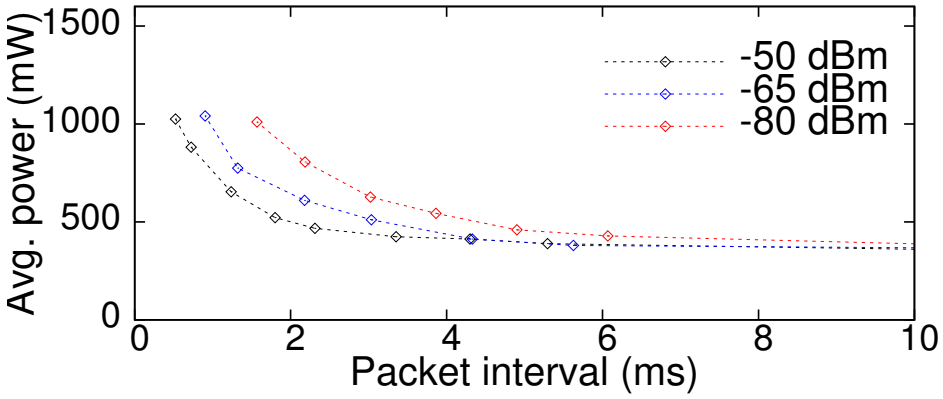
formance of PIMM in various scenarios and compare the accuracy with other existing models in Section 2.5.

2.4 Practical issues

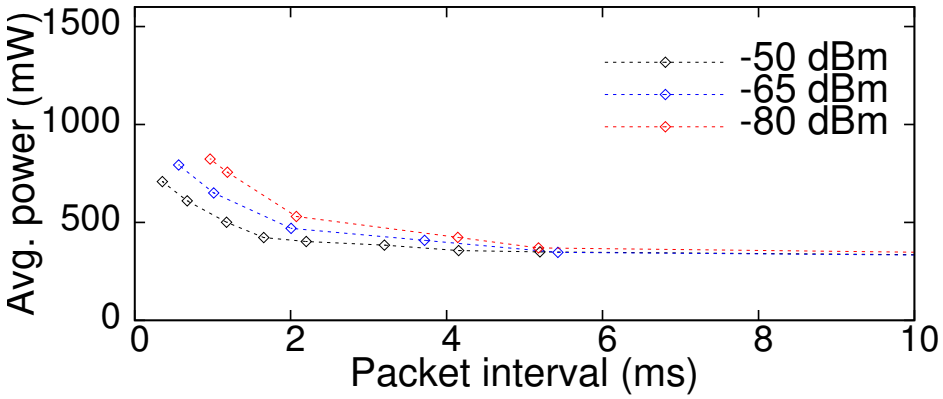
2.4.1 Impact of packet length

Now, we investigate the impact of packet length to the network interface power consumption. In Section 2.3.2, all the packets have 1400-byte application layer payload. This assumption is quite reasonable for file transfer scenarios in which most packets have lengths similar to the maximum transmission unit (MTU), and hence, has been adopted in many existing power models [10, 15, 16, 39]. However, it is not appropriate for other types of traffic, e.g., the audio streaming and online gaming, of which the packet length are often variable. In addition, the packets of VoIP traffic mostly have small application payload, e.g., under 200 bytes.

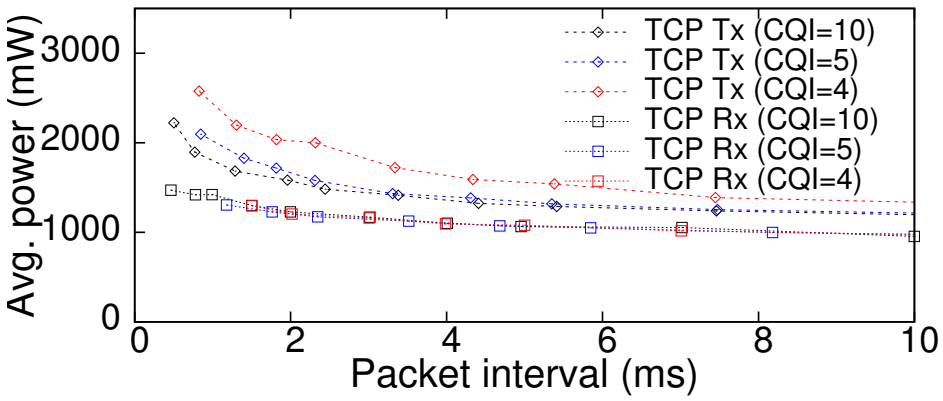
To get the empirical result, we send UDP packets for 20 s with fixed packet length and interval, and measure the average power consumption during the packet transmission. We iterate this experiment 10 times and obtain the average and standard deviation



(a) Wi-Fi TCP tx.



(b) Wi-Fi TCP rx.



(c) LTE TCP tx/rx.

Figure 2.11: Wi-Fi/LTE interface power consumption with respect to the packet intervals for three different levels of channel quality.

of measured power consumption. Fig. 2.9 shows the LTE tx power consumption for each packet interval[4] in $\{1, 2, 4, 6\}$ (ms), with various packet lengths in $\{100, 200, \ldots, 1400\}$ (bytes). As expected, the power consumption linearly increases as the packet length increases for each packet interval under almost the same MCS (modulation and coding scheme) as we experiment at the same place at night when the traffic load is light.

Table 2.4: Coefficients of power function approximation for the 100-byte packet tx/rx power consumption (mW) of network interfaces.

(for t ms)	tx			rx		
$\alpha \cdot t^{\beta}$	α	β	R^2	α	β	R^2
Wi-Fi 2.4	459.54	-0.317	0.95	342.02	-0.213	0.95
Wi-Fi 5	488.26	-0.134	0.96	455.98	-0.172	0.97
LTE	1275.8	-0.049	0.96	1248.2	-0.138	0.95

From the linearity of the power consumption with respect to the packet length, we can estimate the power consumption of the network interfaces for the packet length l with the packet interval t, $P_{net}(t, l)$, by interpolating the two power functions of reference packet lengths, $P_{net}(t, L_1)$ and $P_{net}(t, L_2)$, as follows:

$$P_{net}(t, l) = \frac{(L_1 - l) \cdot P_{net}(t, L_2) + (l - L_2) \cdot P_{net}(t, L_1)}{(L_1 - L_2)}, \tag{2.4}$$

where L_1 and L_2 are reference packet lengths. In this chapter, we set $L_1 = 1400$ and $L_2 = 100$, which represent reasonably long and short packet lengths, respectively. The parameters for Wi-Fi and LTE power model with different packet lengths L_1 and L_2 are summarized in Tables 2.3 and 2.4, respectively.

Fig. 2.10 shows the measured power consumption for $\{100, 400, 700, 1000, 1400\}$-byte packet tx via LTE and the estimated power consumption using Tables 2.3 and 2.4,

[4]The packet interval in Fig. 2.9 is the time interval between the consecutive packet generations at the application layer, while the packet interval in the other figures and text means the time interval over the air.

and (2.4). Furthermore, we additionally conduct the experiment for tx/rx with Wi-Fi and LTE, and the MAPEs (mean absolute percentage error) of the estimated power for {200, 300, ..., 1200, 1300}-byte packet tx/rx via LTE, Wi-Fi 2.4 GHz, and 5 GHz are calculated as 3.25%, 3.58%, and 3.40%, respectively. As a result, we validate the power consumption of arbitrary packet lengths can be properly estimated by interpolating the approximation functions of the reference packet lengths.

2.4.2 Impact of channel quality

As discussed in Section 2.3.1, poor channel quality causes higher energy consumption between consecutive packet interval, and hence, we introduce the parameters α_c and β_c instead of α and β in (2.2) to reflect the channel impact on power consumption as follows:

$$P_{net}(t) = \max(\alpha_c \cdot t^{\beta_c}, P_{tail}), \;\; t < T_{to}. \tag{2.5}$$

Wi-Fi: With low RSS, the packets are transmitted with low-order MCS due to the rate adaptation, thus resulting in increased tx/rx airtime ratios, and the retransmitted packets incur additional energy consumption between consecutive successfully received data packets. Furthermore, much more MAC frame overheads arise when the RSS is below -80 dBm, e.g., active scanning for searching better APs and heavy retransmissions of NDPs due to the receive sensitivity difference between the antenna of APs and that of the smartphones. Figs. 2.11a and 2.11b present the increased per-packet power consumption at the locations of -65 dBm RSS and -80 dBm RSS compared to the location of -50 dBm RSS, which represents good channel quality.

LTE: We set the channel quality indicator (*CQI*)[5] as the metric of LTE channel quality. At low *CQI* region, the smartphone (UE) uses higher transmit power causing

[5]The *CQI* is the UE feedback indicating downlink channel quality and UE's receive capability, and helps eNodeB determine the downlink data rate which can be supported. The UE determines the *CQI* corresponding to the MCS which allows the UE to decode the downlink data with less than 10% error rate.

higher power consumption, and the data from eNodeB would be modulated by lower-order MCS. We select three spots where the different *CQI*s are measured, i.e., *CQI* $= 10$, 5, and 4.[6] We conduct the experiments at these three spots with 1400-byte packets by varying the packet generation interval. The smartphone (UE) and the local server are connected over TCP session to incorporate the retransmission effect for bad channel quality. Fig. 2.11c presents the LTE interface power consumption at the three spots with respect to the average TCP data packet intervals. In the case of TCP tx, the LTE interface consumes more power at the place where the lower *CQI* is reported, i.e, the power consumption at the *CQI* $= 4$ spot is the largest among the three spots, since the eNodeB commands the UE to transmit packets with higher transmit power level. On the other hand, the LTE rx power consumption negligibly varies according to the channel conditions since the eNodeB controls the downlink MCS to reduce retransmissions.

Summary: For both Wi-Fi and LTE interfaces, the *per-packet* power consumption increases as the channel condition becomes poor while the maximum achievable TCP throughput decreases due to low-order MCS, retransmissions, and high transmit power. We generalize the network interface *per-packet* power model at different channel conditions by (2.5). Since the impact of channel quality on LTE rx *per-packet* power is not significant, the ratios of α_c and β_c to α and β (of Tables 2.3 and 2.4) only for Wi-Fi tx/rx and LTE tx are summarized in Table 2.5.

2.5 Performance Evaluation

In this section, we evaluate the accuracy of PIMM in comparison with other existing models and measurement results. For this work, we implement an on-line power estimation algorithm for PIMM in the Android application based on packet traces. We recalibrate all the coefficients of the comparison models [10, 11, 15, 16] fitted to the

[6] In our area, where LTE eNodeBs are seamlessly deployed, we hardly found places with *CQI* under 5.

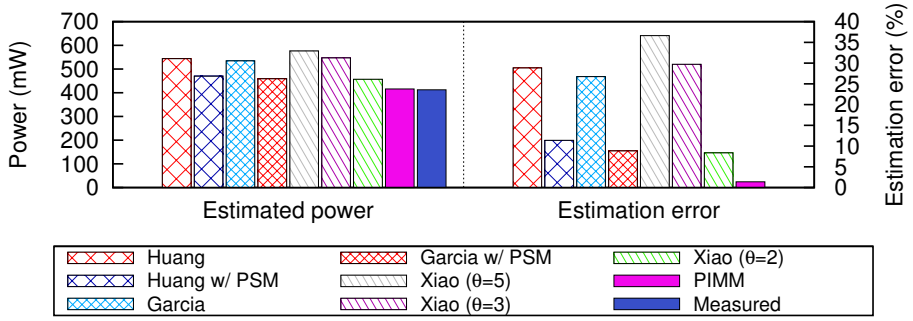
Table 2.5: Channel-aware power function coefficient compensation.

Wi-Fi					LTE		
	tx		rx			tx	
RSS	α_c/α	β_c/β	α_c/α	β_c/β	CQI	α_c/α	β_c/β
-50	1	1	1	1	10	1	1
-65	1.28	1.31	1.21	1.3	5	1.07	1.11
-80	1.75	1.61	1.47	1.62	4	1.32	1.33

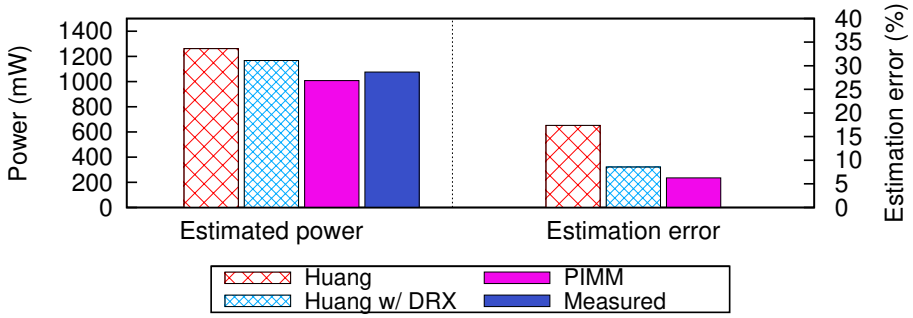
smartphone used in this validation for a fair comparison.

2.5.1 On-line power estimation

We propose the on-line algorithm for PIMM to estimate the average power consumption from packet traces. The timestamp (T), the type of a network interface where the packet passes (N_{type}, either Wi-Fi or LTE), packet length (l), and channel quality (ch) of each packet are obtained from packet traces. Next, set the ***LPM timeout*** t_{tail} and the tail power P_s of N_{type} and calculate packet intervals t^n (equal to t^w for Wi-Fi or t^l for LTE in Fig. 2.7) and t^c, where T^w_{last} and T^l_{last} are timestamps of the last packet of Wi-Fi and LTE interface, respectively, and T^c_{last} is the timestamp of the last packet without distinction of N_{type}. t^n is then divided into t^n_a and t^b_s by comparing t^n and t_{tail} and P_{net} is applied for t^n_a and P_s for t^n_s. P_{net}, the *per-packet* power of N_{type}, is determined by (2.2), (2.4), and t^n with appropriate α and β using channel quality ch and packet length l. P_{proc} is determined by CPU clock frequency f_c, total CPU usage u_{cpu}, and t^c. The *per-packet* energy consumption is cumulatively added to the total energy consumption E, and the average power consumption is finally calculated by dividing E by the total time. The algorithm is described in Algorithm 1.



(a) Wi-Fi.



(b) LTE.

Figure 2.12: Estimated power and estimation error of PIMM and comparison models for 1400-byte UDP tx with random packet intervals using (a) Wi-Fi and (b) LTE.

Algorithm 1 PIMM power estimation algorithm from packet trace

$T^w_{\text{last}} \leftarrow 0, T^l_{\text{last}} \leftarrow 0, T^c_{\text{last}} \leftarrow 0$
for all packets **do**
  $(T, N_{\text{type}}, l, ch) \leftarrow Get\_Packet\_Information()$
  $(t_{tail}, P_s) \leftarrow Set\_Network\_Parameters(N_{\text{type}})$
  if N_{type} = Wi-Fi **then**
    $t^n \leftarrow T - T^w_{\text{last}}$
    $T^w_{\text{last}} \leftarrow T$
  else if N_{type} = LTE **then**
    $t^n \leftarrow T - T^l_{\text{last}}$
    $T^l_{\text{last}} \leftarrow T$
  $t^c \leftarrow T - T^c_{last}$
  $T^c_{\text{last}} \leftarrow T$
  $t^n_a \leftarrow \min(t^n, t_{tail}), t^n_s \leftarrow \max(0, t^n - t_{tail})$
  $P_{net} \leftarrow Cal\_Network\_P_{pp}(ch, l, N_{type}, t^n_a)$
  $P_{proc} \leftarrow Cal\_Proc\_P_{pp}(f_c, u_{cpu}, t^c)$
  $E \leftarrow E + P_{net} \cdot t^n_a + P_s \cdot t^n_s + P_{proc} \cdot t^c$
$P \leftarrow \frac{E}{total\ time}$

2.5.2 Single network data connections

Modeling validation: To validate the accuracy of PIMM for arbitrary traffic pattern, we send the 1400-byte UDP packets by regulating the app-layer packet intervals for 300 s. Every packet is sent with the interval (ms) selected within the range [0, 1000] via Wi-Fi and [0, 15000] via LTE to generate burst traffic and sparse traffic alternately. The smartphone logs the packet intervals during transmission as the input trace of PIMM to estimate the average power consumption. We conduct the experiment in a clean channel at the 5 GHz band for Wi-Fi to avoid interference and congestion. For comparison, we use the model proposed by Huang *et al.* [10] for both Wi-Fi and LTE traffic, the models proposed by Xiao *et al.* [15, 16] and Garcia *et al.* [11] only for Wi-Fi traffic. Even though [39] proposes an LTE power model, we do not compare with

it because it needs the transmit power, received signal strength, and airtime of every packet, which are not available at the smartphone side.

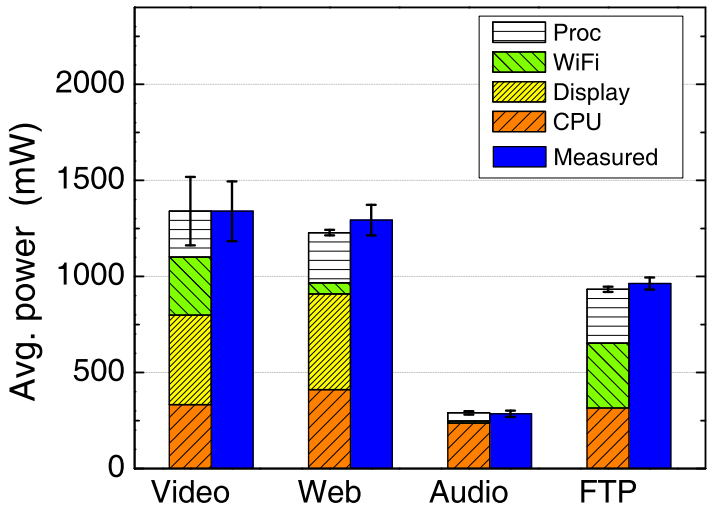
Fig. 2.12 shows the estimated power and estimation error of each model for Wi-Fi (a) and LTE (b). Both the throughput-based model [10] and airtime-based model [11] without considering the PSM/DRX overestimate the power. Although [10] and [11] additionally consider the PSM/DRX, they result in relatively high estimation errors because they do not consider the fact that the CPU packet processing power differs according to the CPU clock frequency. Xiao *et al.* [15, 16] always overestimate the power even though $\theta = 3$ ms is the best threshold in their paper. PIMM reflects the idle time between consecutive packets while Xiao *et al.* [15, 16] ignore the idle time within a packet burst. In summary, PIMM outperforms existing power models for Wi-Fi and LTE by 1) effectively distinguishing active and idle states of network interface, and 2) accurately estimating the energy consumption for inter-packet time with *per-packet* power concept.

Real-world applications: We validate the performance of PIMM with four representative real-world applications, i.e., video streaming, web browsing, audio streaming, and file transfer protocol (FTP). The real-world applications utilize various components and the major power consuming components among them are CPU, display, and network interfaces. Before the model validation, we have run the benchmark application for display power modeling based on the modeling method described in [14] with a fixed brightness level for a simple model. The total smartphone power consumption model is constructed as follows:

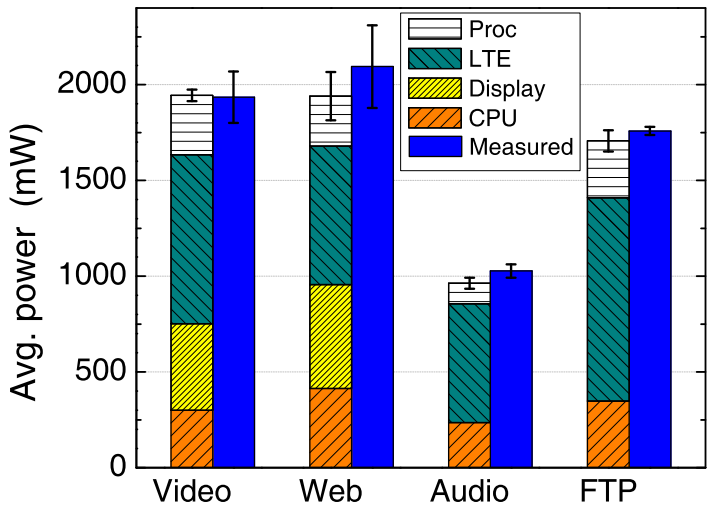
$$P = P_{Wi-Fi} + P_{lte} + P_{proc} + P_{cpu} + P_{display}, \tag{2.6}$$

where P_{proc} is the packet processing power and P_{cpu} is the additional CPU power except for the packet processing power in Section 2.3.2.

We run each application with five trials, and each trial is executed for 60 s. The FTP and audio applications run as background processes, i.e., the display is turned off. For the FTP, the file size is 25 Mbytes, and the average bitrate of streamed videos is



(a) Wi-Fi.



(b) LTE.

Figure 2.13: Estimated power consumption breakdown and measured power consumption for various real-world applications using (a) Wi-Fi and (b) LTE.

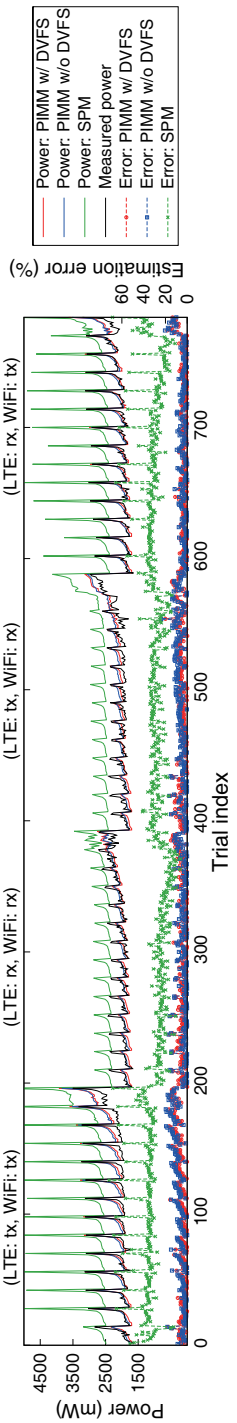


Figure 2.14: Estimated power and estimation error of each trial with parallel TCP connections via both Wi-Fi and LTE.

1.4 Mb/s. The packet traces captured by *tcpdump* are used to estimate P_{Wi-Fi}, P_{lte}, and P_{proc} by Algorithm 1, and the CPU usage and the clock frequency are logged to estimate P_{proc} and P_{cpu}. For the foreground applications, the images displayed on the screen are captured every 3 s, and the display power model converts the pixel information of the captured image into $P_{display}$. The additional power consumption for logging and packet capturing is 85 mW including the image capture, and only 10 mW excluding the image capture. Fig. 2.13 shows the expectations and standard variations of the estimated power consumption by (2.6) and measured power consumption for various real-world applications, and breakdown of the average estimated power consumption, where "Proc" denotes the packet processing power.

In the case of video streaming over Wi-Fi, 41.1% (559 mW) of total power is attributed by the network operation ("Proc"+"Wi-Fi"), which are additionally required for video streaming compared with playing locally stored video. On the other hand, since the LTE consumes more power than Wi-Fi for the same amount of traffic in addition to the longer tail time and higher tail power, 60.6% (1192 mW) of total power (1944 mW) is caused by the network operation ("Proc"+"LTE") in the case of video over LTE. In other words, 71% and 150%[7] more power are additionally consumed for video streaming over Wi-Fi and LTE, respectively, compared with playing local video. In the case of audio streaming, the size of audio data is comparable small, and hence, "Wi-Fi" takes only few portion of average power consumption, while "LTE" takes considerably higher portion for the same reason as the video streaming case. "Wi-Fi" and "LTE" portions of web surfing can vary according to the users' web surfing pattern and the size of Java scripts and images in the web pages. For FTP, "Wi-Fi" and "LTE" portions are related to the Wi-Fi and LTE throughput. Low average power consumption for FTP does not indicate high energy efficiency since the file download completion time should be different according to the link throughput. Overall, PIMM

[7]The values can vary depending on the display brightness level, CPU governor policy, and video bitrate.

with CPU and display model accurately estimates the power consumption for real-world applications with average estimation error of 6.1% for Wi-Fi and 8.6% for LTE.

2.5.3 Multiple network data connections

Modeling validation: We evaluate the accuracy of PIMM for multiple interface-activated scenarios by setting the packet generation rates of two TCP connections to $(\gamma_{lte}, \gamma_{Wi-Fi}) \in \Gamma$ with 1400-byte packets. The set Γ is defined as $\Gamma = \Gamma_{lte} \times \Gamma_{Wi-Fi} = \{(x, y) | \ x \in \Gamma_{lte} \wedge y \in \Gamma_{Wi-Fi}\}$ by the cartesian product of Γ_{lte} and Γ_{Wi-Fi}, where $\Gamma_{lte} = \Gamma_{Wi-Fi} = \{10,$ 20, 50, 100, 111, 125, 143, 167, 200, 250, 333, 500, 1000, full$\}$ (packets/s), such that $|\Gamma| = 196$. For $\gamma = \text{full}$, the sender tries to send packets as fast as possible. The packet generation rates limit the maximum TCP throughput and the actual TCP throughput is determined by round trip time and packet loss pattern of that TCP link. We firstly let the smartphone act as TCP tx via Wi-Fi and LTE, and then repeat the experiments after changing the direction of TCP at the smartphone to (LTE: rx, Wi-Fi: rx), (LTE: tx, Wi-Fi: rx) and (LTE: rx, Wi-Fi: tx). Especially, (LTE: rx, Wi-Fi: tx) and (LTE: tx, Wi-Fi: rx) represent the case of smartphones providing a mobile hotspot, which is one of the most popular applications utilizing both Wi-Fi and LTE interfaces simultaneously.

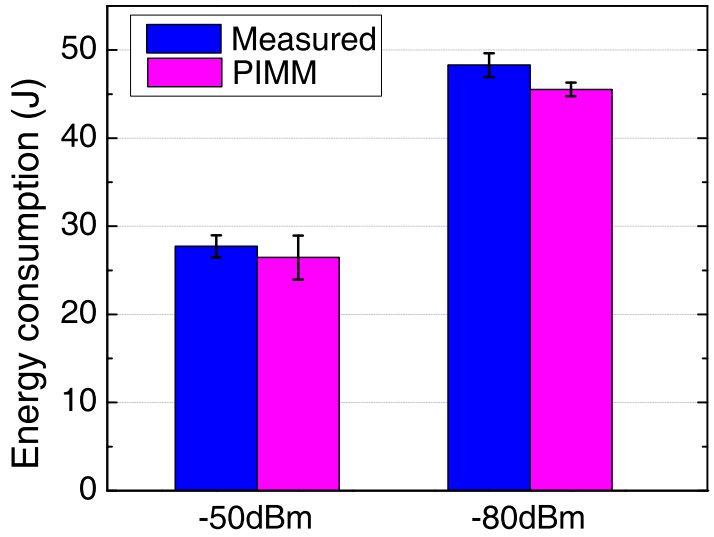
We compare the modeling accuracy of PIMM with SPM, representing the sum of Wi-Fi and LTE power models, as proposed in [37, 38]. For SPM, the individual power model for data transfer via Wi-Fi or LTE is obtained based on the linear fitting function of the Wi-Fi and LTE throughput variables x_w and x_l, represented as $P_{Wi-Fi} = (a_w x_w + b_w)$ and $P_{LTE} = (a_l x_l + b_l)$, where $\{a_w, b_w\}$ and $\{a_l, b_l\}$ are the linear fitting coefficients for the Wi-Fi and LTE throughput-power curves, respectively. Each power model involves the packet processing power according to the throughput variable. Then, SPM is obtained as the sum of the two linear functions, i.e., SPM $= P_{Wi-Fi} + P_{LTE}$.

Fig. 2.14 shows the estimated power and estimation error of the PIMM and SPM for the parallel TCP connections via Wi-Fi and LTE. In summary, "PIMM with DVFS"

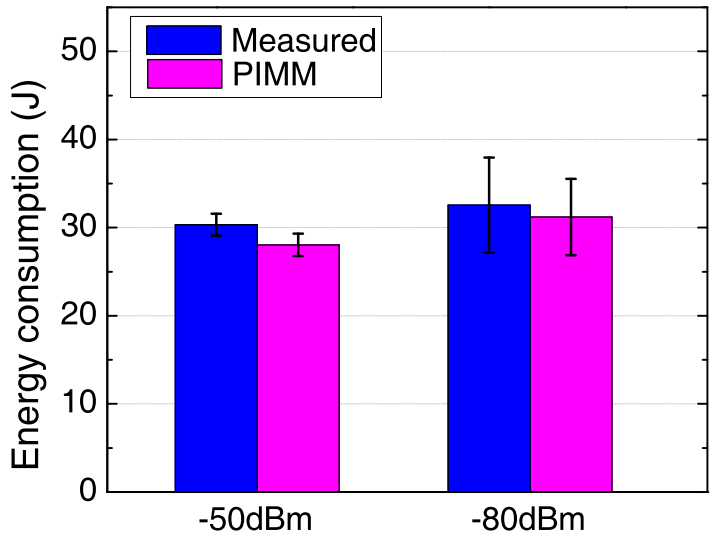
shows the best result of the power estimation ($4.9 \pm 3.7\%$ MAPE). The estimation error slightly increases to $5.4 \pm 4.3\%$ MAPE without DVFS. However, "SPM" shows much higher estimation error ($29.9 \pm 9.0\%$) for the parallel TCP connections via Wi-Fi and LTE. The reason is that the SPM is the sum of $P_{Wi-Fi} + P_{proc}$ ($= P_{Wi-Fi}$) and $P_{lte} + P_{proc}$ ($= P_{LTE}$), thus incorrectly summing P_{proc} twice, and hence, results in the overestimation of the power by b in Table 2.2 compared to PIMM.

FTP using both Wi-Fi and LTE: Here, we investigate the estimation accuracy of PIMM for the FTP scenarios, especially the smartphone downloads the segments of the file using both Wi-Fi and LTE interfaces. We emulate the FTP for this evaluation by activating both Wi-Fi and LTE TCP connections as presented in Section 2.3.2 with two local servers. 25 Mbyte-file download is considered with two downloading policies; with "Policy 1," the file is equally divided into two segments, and each segment is downloaded via Wi-Fi and LTE. With "Policy 2," packets are downloaded via Wi-Fi and LTE until the cumulative received data size becomes 25 Mbytes, i.e., the data transfer times for Wi-Fi and LTE are the same for both.

Fig. 2.15 shows the measurement and estimation results at two locations, namely "-50 dBm" and "-80 dBm", where the LTE channels are good while the Wi-Fi (2.4 GHz band) channels are good (-50 dBm RSS) and bad (-80 dBm RSS), respectively. The average throughput of LTE is 12.8 Mb/s, and Wi-Fi throughput at the good and the bad channels are 5.1 Mb/s and 1.4 Mb/s, respectively. PIMM estimates the energy consumption with the error of $5.7 \pm 1.7\%$. For "Policy 1", the energy consumption highly increases due to the low Wi-Fi throughput at "-80 dBm" even though the LTE already finished the downloading. On the other hands, for "Policy 2", the results show the similar energy consumption since the downloaded data size via the LTE is dominant for the both locations. However, "-80 dBm" consumes slightly more energy than "-50 dBm" due to the increase of the LTE download time and the increased power consumption of Wi-Fi at low RSS. For "-50 dBm" cases, "Policy 1" is more efficient than "Policy 2" due to short LTE download time since the LTE throughput is



(a) Policy 1.



(b) Policy 2.

Figure 2.15: Measured and estimated energy consumption of downloading 25 Mbtyes file using both Wi-Fi and LTE TCP connections.

even higher than that of Wi-Fi, but the lower energy efficiency (J/bit) of LTE.

To investigate the energy efficiency of FTP using both Wi-Fi and LTE, we measure the energy consumption of downloading 25 Mbyte-file only via LTE ("Only LTE") and the averaged result is 32.9 J. The energy efficiency improvement ratios to "Only LTE" are 15.8% ("Policy 1") and 7.9% ("Policy 2") at "-50 dBm", and 1.1% ("Policy 2") at "-80 dBm", but the efficiency of "Policy 1" at "-80 dBm" decreases by 46.5% compared to "Only LTE". From these observations, we find that utilizing Wi-Fi and LTE links with proper network selection policy and download file segment allocations enhances the energy efficiency of the smartphones, and PIMM helps estimate the performance with expected packet traces.

2.5.4 Model generation complexity

Let (N_n, N_l, N_q, N_i) be a 4-tuple denoting the cases to be modeled for network interface modeling, where the four elements are the numbers of network interfaces, packet lengths, signal quality levels, and packet intervals, respectively. The overall model generation complexity becomes $O(N_n \cdot N_l \cdot N_q \cdot N_i)$. In our modeling procedure, the measurements have been conducted for $N_n = 2$ (Wi-Fi and LTE), $N_l = 2$ (1400 and 100-byte packets), $N_q = 3$ (-50,-65, and -80 dBm for Wi-Fi and CQI $= 4$, 5, and 10 for LTE), and $N_i = 10$ (from full-pumping to light load traffic). For CPU packet processing power modeling, $N_c \cdot N_i$ additional measurements are needed, where N_c is the number of CPU clock frequencies to be modeled.

We obtain the PIMM model of SHV-E330 smartphone with the same modeling procedure above, and the average R^2 (in Table 2.3) of the *power function* approximation of the network interfaces is $0.95(\pm 0.02)\%$, and estimation errors for FTP scenarios in Sections 2.5.2 and 2.5.3 are 3.5% for Wi-Fi, 4.2% for LTE, and 4.7% when both Wi-Fi and LTE are activated. Accordingly, we validate that PIMM accurately estimates the power consumption of SHV-E330 in addition to that of SHV-E120 smartphone.

2.6 Summary

In this chapter, we propose a packet interval-based power model for multiple network interface-activated smartphones, which estimates the average power consumption based on the packet interval. The proposed power model is constructed by combining *per-packet* power of the packet processing, Wi-Fi, and LTE network interfaces. The accuracy of our model is evaluated in various scenarios including the real-world smartphone applications. We validate that our model outperforms other existing power models for single/multiple network transmissions in terms of the estimation error.

We prospect that PIMM will help estimate the future energy consumption as part of the energy-aware algorithm operation and develop energy-efficient protocols/algorithms especially utilizing multiple network interfaces. PIMM can be used in a simulator as a power model, which are not possible with measuring power consumption using a real hardware. As future work, we will further evaluate the *download booster* and *Airplug* with PIMM.

Chapter 3

BattTracker: Enabling Energy Awareness for Smartphone Using Li-ion Battery Characteristics

3.1 Introduction

Enabling battery drain monitoring is very important to manage the remaining energy for battery-limited mobile devices such as smartphones. In order to manage battery lifetime, previous studies in [40, 41] try to provide detailed battery status to users and application developers. In addition, for the purpose of energy consumption monitoring, previous studies have modeled the power consumption of smartphones [12–14, 19, 20, 42, 43]. Most approaches have used an external power measurement tool to measure and model the power consumption of each component inside smartphone, e.g., CPU, display, and network interfaces [12–14, 42, 43]. Accordingly, they need extensive training to obtain accurate power model. Instead of using such a measurement tool, the information reported by a battery fuel gauge [44, 45] can be used to obtain discharge current drawn by a device [19, 20]. In [20], the discharge current is estimated by dividing the resistive voltage drop of a battery by the battery's internal resistance, and the authors use the estimated discharge current to construct the power model of a smartphone.

Meanwhile, even if devices consume the same amount of power, battery drain rate and lifetime[1] can vary according to their battery capacity. Furthermore, battery characteristics such as capacity and internal resistance vary according to temperature and the degree of battery aging [1, 2]. For these reasons, even though the existing power modeling-based methods accurately estimate energy consumption, they fail to provide accurate battery lifetime in varying temperature or for a device powered by an aged battery. The approach in [20] also results in higher estimation error in a similar situation unless the impact of temperature and aging on battery internal resistance is not considered. Therefore, the impact of temperature and aging on battery capacity and resistance should be studied and reflected if we exploit them to estimate battery drain rate/lifetime. Unfortunately, it is difficult to quantify the degree of battery aging for a given Li-ion battery without any helpful information such as cumulative charging/discharging cycles. This limitation is one of the obstacles for us to model the impact of battery aging on battery characteristics.

As another approach to estimate battery lifetime, State-of-Charge (SoC), which is provided by a battery fuel gauge, can be used. Since the granularity of SoC is 1% of battery capacity, it does not achieve high enough time resolution to realize real-time battery drain rate monitoring [13].

To enable instantaneous battery drain monitoring taking temperature and battery aging into consideration, we propose *BattTracker*, a novel battery drain rate/lifetime estimation algorithm for mobile devices in real-time using battery characteristics. Specifically, we design *BattTracker* to achieve the following goals: (a) it accurately provides instantaneous battery drain rate, (b) it estimates battery drain rate considering both temperature and battery aging, and (c) most importantly, we develop a solution which does not require any training effort to figure out the detailed impact of temperature and battery aging on battery characteristics.

[1] Battery lifetime is convertible to battery drain rate, which represents how fast battery energy is consumed.

With instantaneous battery drain rate estimated by *BattTracker*, Users can manage their battery lifetime by themselves and mobile application developers can embed energy-awareness into their applications such as video streaming, web surfing, and file transferring to optimize energy efficiency or to guarantee the battery lifetime. For example, a video streaming application with dynamic adaptive streaming over HTTP (DASH) can adaptively request a video with the optimal source rate in terms of energy efficiency within a required battery lifetime by being aware of battery drain rate.

Our contributions are summarized as follows:

- Through measurement, we model the effects of temperature and aging on capacity and internal resistance of Li-ion batteries used in smartphones.
- We propose the effective resistance concept, which helps estimate battery drain rate without knowing the detailed impact of temperature and aging on battery characteristics.
- We propose *BattTracker*, which estimates battery drain rate during run time for any temperature and any degree of battery aging with fine-grained time resolution.
- The performance of *BattTracker* is extensively evaluated with smartphones and differently aged batteries for various mobile applications with varying temperature, and confirm that *BattTracker* estimates battery drain rate with high accuracy.

The rest of this chapter is organized as follows. Section 3.2 provides the background for battery interface and Li-ion battery. Section 3.3 describes Li-ion battery characteristics. The concept of effective resistance is proposed in Section 3.4. The design and key algorithms of *BattTracker* are presented in Section 3.5. Section 3.6 evaluates the performance of the *BattTracker*. Section 3.2.4 presents related work, and the chapter concludes in Section 3.7 with future work.

3.2 Background

3.2.1 Smartphone's battery interface

Smartphone's battery interface, called battery monitoring unit (BMU) [46] in battery-powered devices, measures battery status such as voltage, temperature, and remaining battery level. Such battery status is stored in registers and system files [19, 20, 44–46]. Android operating system (OS) can read the registers and utilize the battery status in order to prevent battery from being overcharged or completely discharged. Smartphones' users can check remaining battery energy from the battery level reported by the battery interface. Android applications can obtain the battery status using *BroadcastReceiver* component of Android, typically every 10 sec. But, it might not be of sufficiently high granularity for certain applications [19, 20]. To obtain battery status with shorter update period, the applications can directly read system files storing the battery status via *cat* command. For example, *cat /sys/class/battery/voltage_now* is used to read the voltage across a device V_{out} in Fig. 3.1.

3.2.2 State of Charge (SoC) and Open Circuit Voltage (V_{oc})

State of Charge (SoC)

Android OS utilizes State-of-Charge (SoC) with the integer values ranging from 0% to 100% to notify smartphone users of the remaining battery level via a notification bar of the smartphone screen. SoC denotes the ratio of the remaining battery capacity to the fully-charged battery capacity, and it is estimated by a battery fuel gauge [44], e.g., MAX incorporation's MAX17050 in Samsung Galaxy S3 (SHV-E210) and Galaxy S4 (SHV-E300). The capacity of a fully-charged Li-ion battery can be different from a value in a battery specification, since the battery capacity is affected by temperature and the degree of battery aging which is referred to as State-of-Health (SoH) [1, 47–49]. SoH is estimated by counting the cumulative number of fully charging/discharging cycles of the battery, and battery capacity decreases as much as the degree of aging.

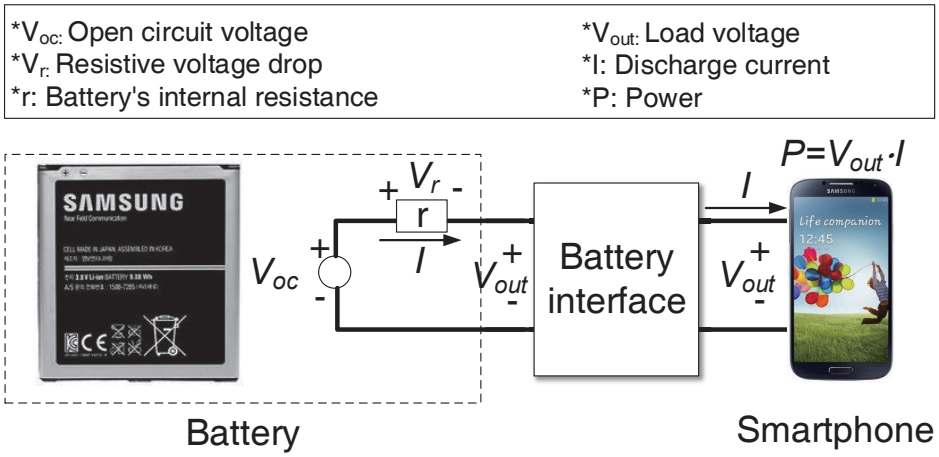
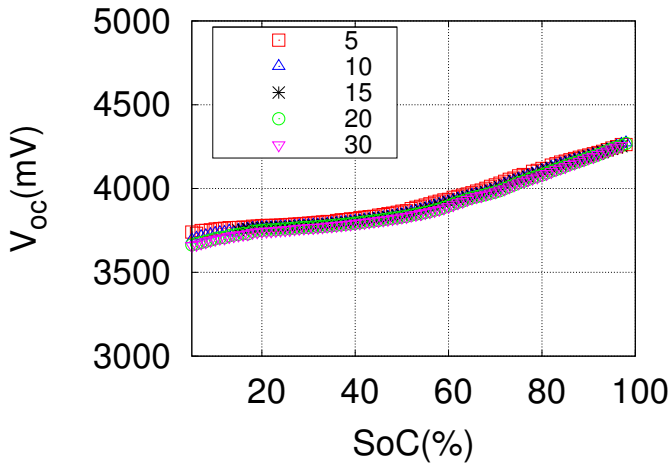


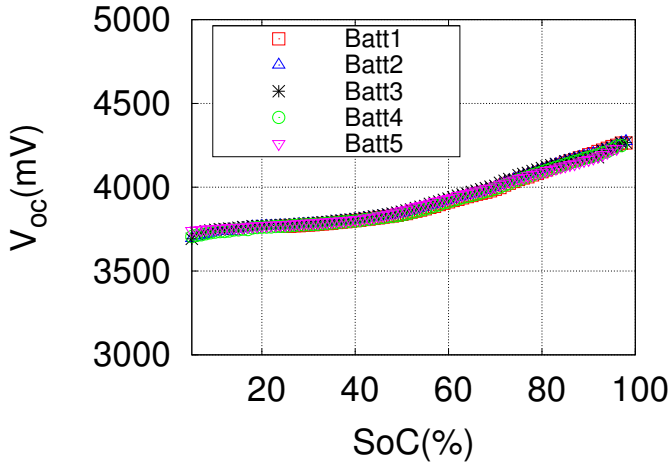
Figure 3.1: The equivalent circuit of a battery-powered mobile device.

Open circuit voltage (V_{oc})

Open circuit voltage (OCV), V_{oc}, is the input voltage of the equivalent circuit of a battery-powered mobile device as illustrated in Fig. 3.1. Even though the specification of a battery describes its input voltage as a representative value, e.g., $3.8\ V$, V_{oc} actually varies from $4.3\ V$ to $3.65\ V$ according to SoC. V_{oc} has a one-to-one relationship with SoC [2, 47], and it is used to infer the remaining battery energy [13, 50]. V_{oc} vs. SoC curves can be obtained by measuring V_{out} and SoC of a smartphone, which stays in an idle state. For the smartphone in the idle state, discharge current (I) is only a few milliwatts, and hence, V_{oc} approximates V_{out}, i.e., $V_{out} = V_{oc} - I \cdot r \simeq V_{oc}$, where r is the smartphone battery's internal resistance. Fig. 3.2 shows changes of V_{oc} with respect to SoC at various temperatures for differently aged batteries. Fig. 3.2a represents V_{oc} vs. SoC of a fresh battery for SHV-E210 at various temperatures where the values in the legend represent the temperature ($°C$), and Fig. 3.2b represents V_{oc} vs. SoC of five differently aged batteries for SHV-E210 at $20\ °C$. We observe that the relationship between V_{oc} and SoC is not affected by the temperature and the battery aging.



(a) Various temperatures.



(b) Various degrees of aging.

V_{oc} vs. SoC curve of SHV-E210 batteries.

3.2.3 Battery's internal resistance

The battery's voltage is not fully delivered to a device due to the battery's internal resistance. In Fig. 3.1, the voltage V_{out} actually delivered to the device is smaller than V_{oc} because of the resistive voltage drop, V_r. According to Ohm's law, V_r is proportional to the current (I) drawn by the device. Similar to the battery capacity, the battery's internal resistance is also affected by temperature and the degree of battery aging.

3.2.4 Related Work

The battery information delivered by the battery interface can be used to model the power consumption of smartphones [19,20]. The authors of [19] propose a self-modeling using the discharge current provided by the battery interface. However, battery interfaces in state-of-the-art smartphones rarely provide the discharge current [20], and hence, it is not applicable to most smartphones. V-edge [20] utilizes resistive voltage drop at the internal resistance of the Li-ion battery to realize the fast power consumption modeling for smartphones. In this case, the value of the internal resistance should be known to convert voltage drop into the corresponding discharge current. However, battery resistance varies according to temperature and battery aging as described in Section 3.3.

The authors of [13] utilize SoC to construct smartphone's power consumption models. Instead of SoC, V_{oc} is used to reduce the measurement time [21] using the relationship between V_{oc} and SoC. V_{oc} provides in average 6.5 times finer granularity than the SoC since V_{oc} varies from 4.3 V to 3.65 V with the unit of 1 mV while SoC varies from 100% to 0%. In addition, the energy loss due to the internal resistance of a Li-ion battery is considered for accurate estimation of remaining battery energy and power modeling of smartphone components [21].

The battery aging of Li-ion batteries for mobile devices is studied in [22]. The authors proposed the method to estimate the degree of aging of an arbitrary Li-ion battery by comparing charging speed of an aged battery to that of a fresh battery.

3.3 Li-ion Battery Characteristics

In this section, we study characteristics of Lithium-ion (Li-ion) battery, mainly focusing on the impact of temperature and aging on battery capacity and internal resistance.

3.3.1 Impact of temperature and aging on Li-ion battery

Li-ion battery characteristics such as capacity and internal resistance are affected by the temperature and aging [1,2]. At low temperature, especially, the capacity decreases due to the degradation of chemical reactions. In addition, the internal resistance increases at low temperature for the same reason. Because of these characteristics, the available battery energy is rapidly depleted at low temperature, e.g., at ski resorts in winter. Similar to low temperature, the battery capacity decreases and the internal resistance increases as the battery ages during its lifetime [1,2].

In summary, both low temperature and aging decrease capacity, and increase internal resistance of a Li-ion battery. However, the previous studies [13,20], which exploit Li-ion battery characteristics to estimate energy consumption, lack the consideration of the temperature and aging effect on the Li-ion battery. If a smartphones is powered by an aged battery and/or exposed at low temperature, the battery status such as power consumption, remaining lifetime, and battery drain rate cannot be accurately estimated with the previous approaches. Therefore, variations of battery capacity and internal resistance should be reflected for accurate battery status estimation.

3.3.2 Measuring battery characteristics

We experimentally investigate how temperature and aging affect the Li-ion battery characteristics. First, we implement an Android application for training, which runs a set of floating point operations as fast as possible with 100% CPU utilization on smartphones.[2] This training application runs floating point operations for T_t (training

[2]The default CPU governor, **On Demand**, is used.

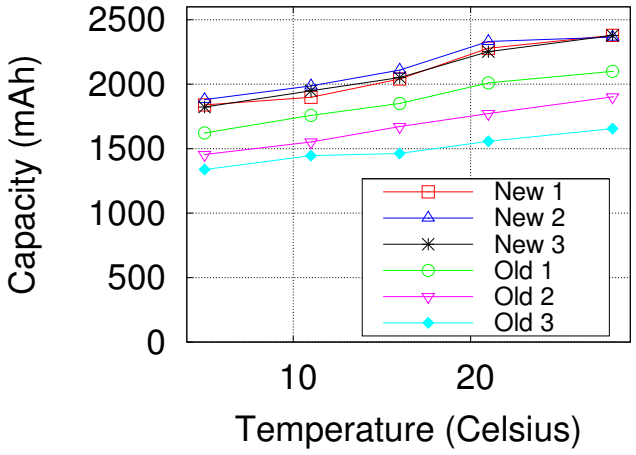
session) followed by a break for T_b (break session), and iterates them until the battery is fully depleted. During the training, V_{out}, temperature, and SoC are logged for every sampling period, T_s. The battery capacity (C) is estimated by $C = E_{iter} \cdot n_{iter}$, where E_{iter}[3] is the energy consumption during one training and break session, and n_{iter} is the number of session iterations until the battery is totally depleted. Then, the internal resistance (r) can be estimated as follows:

$$r = \frac{\overline{V_b} - \overline{V_t}}{(P_t/\overline{V_t} - P_b/\overline{V_b})}, \tag{3.1}$$

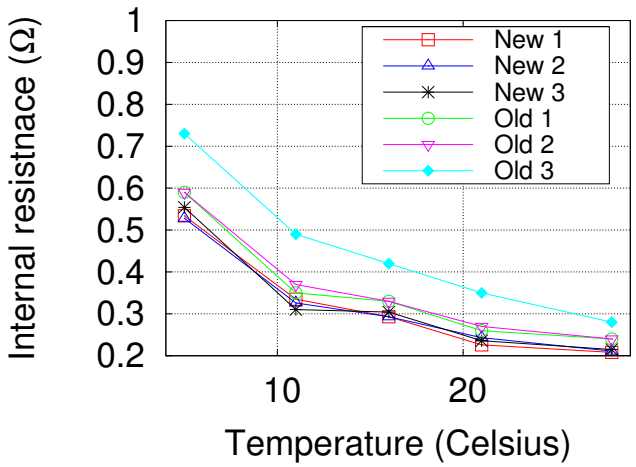
where r is the internal resistance, and $\overline{V_t}$ and $\overline{V_b}$ are the average V_{out} measured during a training and break session, respectively. The difference between $\overline{V_b}$ and $\overline{V_t}$ is caused by the difference between resistive voltage drops during training and break sessions. On the other hand, $(P_t/\overline{V_t} - P_b/\overline{V_b})$ represents the difference between the discharge currents during training and break sessions.

We conducted experiments with three different fresh batteries and three differently aged batteries of SHV-E210 smartphone. In this experiment, T_t, T_b, and T_s are set to 22 sec, 17 sec, and 2 sec, respectively, and P_t and P_b are 0.9 W and 0.09 W, respectively. The smartphone is placed in a temperature controlled water bath to maintain the temperature of the batteries during the training. Fig. 3.3 shows the estimated capacity and internal resistances of six batteries at five different temperatures. Each value of temperature in Fig. 3.3 is the average temperature measured by smartphone's battery interface during each training. From Fig. 3.3, we observe that 1) the fresh batteries of the same model have similar values of capacity and internal resistance for various temperatures, and 2) capacity and internal resistance of an aged battery have a scaling relationship to those of a fresh battery.

[3] $E_{iter} = P_t \cdot T_t + P_b \cdot T_b$, where P_t and P_b are the average power consumption of training and break sessions, respectively.



(a) Capacity vs. temperature



(b) Internal resistance vs. temperature

Measured capacity and internal resistance of SHV-E210 batteries with respect to the temperature.

3.3.3 Battery characteristics models

Based on the observations from the measurements, we model battery characteristics, i.e., the capacity and the internal resistance. Let C_f and r_f be the capacity and the internal resistance of a 'fresh' battery at the room temperature (20 °C), respectively. Considering the observations, we model the battery capacity and internal resistance of an aged battery having d_A degree of aging at temperature T, denoted by $C(T, d_A)$ and $r(T, d_A)$, with scaling parameters, μ and ε, as follows:

$$C(T, d_A) = \varepsilon_t(T) \cdot \varepsilon_a(d_A) \cdot C_f, \tag{3.2}$$

$$r(T, d_A) = \mu_t(T) \cdot \mu_a(d_A) \cdot r_f, \tag{3.3}$$

where ε_t, μ_t, ε_a, and μ_a denote the effect of temperature on capacity and resistance, and that of aging on them, respectively. ε_t and μ_t are modeled as piecewise linear functions of temperature. On the other hand, ε_a and μ_a of a certain aged battery are obtained by normalizing the capacity and internal resistance by those of the fresh battery, respectively.

Table 3.1 shows the averages and standard deviations of both ε_a and μ_a of three aged batteries, obtained from the samples measured at five different temperatures. We observed that small standard deviations of both ε_a and μ_a across three batteries, implying that temperature's influence on ε_a and μ_a is minimal. Therefore, the effect of temperature and that of aging on capacity and resistance are independently modeled with ε's and μ's, respectively, as (3.2) and (3.3). Since μ_a increases and ε_a decreases as a battery suffers from aging, the parameter $\frac{\mu_a}{\varepsilon_a}$ can represent the degree of battery aging. Even though we do not know the exact degrees of aging of the three batteries, we infer that *Old 1* is the least aged among the three since its $\frac{\mu_a}{\varepsilon_a}$ value is only 1.25 while the other batteries' values are 1.42 and 2.00.

In summary, we can model the effects of temperature and battery aging on the battery characteristics as (3.2) and (3.3) assuming that they affect the battery characteristics independently based on the results in Table 3.1.

Table 3.1: Average μ_a, ε_a, and μ_a/ε_a of three aged batteries for SHV-E210 at five different temperatures.

	Old 1	Old 2	Old 3
μ_a	1.11±0.02	1.13±0.02	1.42±0.07
ε_a	0.89±0.01	0.79±0.01	0.71±0.02
μ_a/ε_a	1.25	1.42	2.00

3.4 Effective Internal Resistance

To model the battery characteristics at various temperatures, substantial training time is needed. Furthermore, it is difficult to model the characteristics considering the battery aging since it requires additional training overhead whenever the models are updated. The authors of [47] experimentally measured the capacities and resistances of various aged batteries, e.g., fresh, 50-days, 100-days, and 150-days batteries. In practice, however, determining the degree of aging is rarely possible for commercial devices.

For these reasons, we propose the concept of effective internal resistance, r_e, to remove the burden of training to update battery characteristics considering the effects of temperature and battery aging on them. r_e is useful to estimate battery drain rate/lifetime since we can estimate them without knowing the exact values of battery capacity and internal resistance by introducing r_e.

3.4.1 Effective internal resistance (r_e)

The effective internal resistance $r_e(T, d_A)$ is defined by the product of the resistance r_f of a fresh battery at room temperature and the scaling parameters ε's and μ's.

$$r_e(T, d_A) \triangleq \varepsilon_t(T) \cdot \varepsilon_a(d_A) \cdot \mu_t(T) \cdot \mu_a(d_A) \cdot r_f, \tag{3.4}$$

where ε's and μ's are the same as those in (3.2) and (3.3). As shown in (3.4), r_e is a scaled value of the present internal resistance, $r(T, d_A)$ ($= \mu_t(T) \cdot \mu_a(d_A) \cdot r_f$), with

Symbol	Description	Model (Equation)
T	Temperature (^oC)	
t_s	Sampling period (sec)	
t_k	k-th sample time (sec)	$k \cdot t_s$
$S[t_k]$	SoC measured at t_k	
$t^{(n)}$	Sample time when SoC changes from $(n+1)\%$ to $n\%$ (sec)	$t^{(n)} = \min\{t_k \mid S[t_k] = n\}$
$\Delta t^{(n)}$	Time duration while SoC is $n\%$ (sec)	$t^{(n-1)} - t^{(n)}$
$C(T, d_A)$	Capacity of d_A-degree aged battery at temperature T (mAh)	(3.2)
$r(T, d_A)$	Resistance of d_A-degree aged battery at temperature T (Ω)	(3.3)
$r_e(T, d_A)$	Effective resistance of d_A-degree aged battery at temperature T (Ω)	(3.4)
$V_{oc}[t_k]$	Open circuit voltage estimated at t_k (mV)	(3.9)
$V_{out}[t_k]$	Voltage across a device measured at t_k (mV)	
$I[t_k]$	Instantaneous discharge current drawn by a device at t_k (mA)	$\frac{(V_{oc}[t_k] - V_{out}[t_k])}{r(T, d_A)}$
$L[t_k]$	Remaining battery lifetime estimated at t_k (h)	$\frac{C(T, d_A) \cdot S[t_k]}{I[t_k]}$
$R_d[t_k]$	Battery drain rate estimated at t_k (%/h)	$\frac{100}{C(T, d_A)/I[t_k]}$
$R^{(n)}$	Average battery drain rate during $\Delta t^{(n)}$ (%/h)	$\frac{1}{\Delta t^{(n)}}$

Table 3.2: Important notations.

the scaling factor equal to $\varepsilon_t(T)\cdot\varepsilon_a(d_A)$. Therefore, r_e is proportional to the current internal resistance $r(T, d_A)$ as well as the capacity scaling factor for temperature and aging. The benefit by adopting r_e is that the exact values of $C(T, d_A)$ and $r(T, d_A)$ are not necessarily required to estimate battery drain rate and lifetime. The details are explained in the following section.

3.4.2 Battery drain rate and lifetime derived from r_e

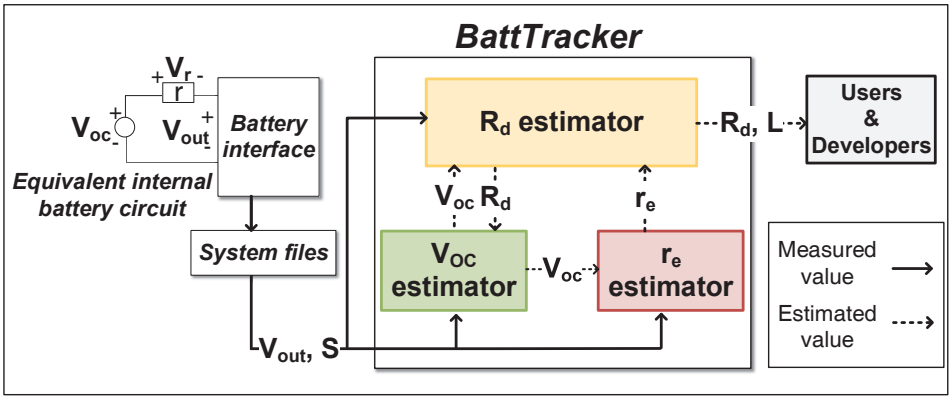
Battery drain rate denotes how fast battery energy is consumed. In this chapter, specifically, the unit of battery drain rate is $\%/h$, e.g., $3\%/h$ drain rate means that 3% of capacity[4] is consumed per hour. Remaining lifetime has a linear relationship with the reciprocal of the battery drain rate. Considering the effect of temperature and aging, we formulate the equations of battery drain rate R_d and remaining lifetime L at time t as follows:

$$\begin{aligned} R_d(t) &= \frac{100}{C(T,d_A)/I(t)} = \frac{100}{C(T,d_A)/(\frac{V_{oc}(t)-V_{out}(t)}{r(T,d_A)})} \\ &= \frac{100\cdot(V_{oc}(t)-V_{out}(t))}{C(T,d_A)\cdot r(T,d_A)} = \frac{100\cdot(V_{oc}(t)-V_{out}(t))}{\varepsilon_t\cdot\varepsilon_a\cdot C_f\cdot\mu_t\cdot\mu_a\cdot r_f} \\ &= \frac{100\cdot(V_{oc}(t)-V_{out}(t))}{C_f\cdot r_e(T,d_A)}, \end{aligned} \tag{3.5}$$

$$L(t) = \frac{C(T,d_A)\cdot S(t)}{I(t)} = (100/R_d(t))\cdot S(t), \tag{3.6}$$

where $S(t)$ is SoC at time t, μ's and ε's are scaling parameters. T and d_A are also time-varying. C_f is the capacity of a fresh battery at the room temperature, but it can be any constant value. As shown in (3.5) and (3.6), R_d is calculated with V_{oc}, V_{out}, and r_e, and L additionally needs S since it requires remaining battery energy. Accordingly, r_e enables us to estimate R_d and L without knowing $r(T, d_A)$ and $C(T, d_A)$. Knowing that V_{out} and SoC can be obtained from the battery interface, we now need to develop the methods to obtain V_{oc} and r_e.

[4]The unit of capacity in this chapter is mAh.



Block diagram of BattTracker. The dashed lines represent the estimated values by each component.

3.5 BattTracker Design

In this section, we introduce the design of *BattTracker* and present the components of *BattTracker* to achieve the following goals: (a) to estimate instantaneous R_d, (b) to track V_{oc} during run time, (c) to estimate r_e to consider temperature and battery aging. The update period of the estimated values, i.e., R_d, V_{oc}, and r_e by *BattTracker* is determined by the sampling period of V_{out} and S. To achieve high sampling frequency of V_{out} and S, *BattTracker* accesses system files and obtains them. For example, the minimum update period of V_{out} and S is 0.5 second in case of MAXIM fuel gauge chip used in SHV-E210, SHV-E300, and SHV-E330 [44]. Fig. 3.4 shows that the overall framework of *BattTracker* with three major components, i.e., 1) R_d *estimator*, 2) V_{oc} *estimator*, and 3) r_e *estimator*. We next explain the detailed operation of each component, and the notations used in Sections 3.5 and 3.6 are summarized in Table 3.2.

Algorithm 2 V_{oc} estimator algorithm

Initialize:

1: $V_{oc} \leftarrow f(S)$

During run time: BattTracker is running SoC changes to n $(n \in 1, 2, 3, \cdots, 99)$

2: $S \leftarrow n,\ V_{oc} \leftarrow f(n)$ $\{V_{oc}$ recalibrator$\}$ $V_{out} \geq V_{oc}$

3: $V_{oc} = V_{out}$ $\{V_{oc}$ recalibrator$\}$

4: Get estimated R_d from R_d *estimator*

5: $V_{oc} \leftarrow UpdateV_{oc}(V_{oc}, S, R_d, t_s)$ $\{V_{oc}$ tracker$\}$

3.5.1 R_d estimator

R_d *estimator* estimates R_d and L using V_{oc} and r_e. At the k-th sample time t_k, R_d and L are calculated based on (3.5), (3.6), and Table 3.2 as follows:

$$\begin{aligned} R_d[t_k] &= 100 \cdot (V_{oc}[t_k] - V_{out}[t_k])/(C_f \cdot r_e), \\ L[t_k] &= (100/R_d[t_k]) \cdot S[t_k]. \end{aligned} \tag{3.7}$$

Since V_{oc} and r_e are time-varying parameters, R_d *estimator* obtains updated values of V_{oc} and r_e from V_{oc} *estimator* and r_e *estimator*, respectively. $R_d[t_k]$, which is estimated by R_d *estimator*, is in turn delivered to V_{oc} *estimator* and r_e *estimator* to update V_{oc} and r_e, respectively.

3.5.2 V_{oc} estimator

R_d *estimator* requires V_{oc} to estimate R_d using (3.5). The main role of V_{oc} *estimator* is tracking V_{oc} variation over time, since V_{oc} is not provided directly by most battery interfaces [20]. V_{oc} *estimator* utilizes the fact that V_{oc} variation follows V_{oc}-SoC curve as shown in Fig. 3.2 regardless of temperature and battery aging [50]. V_{oc} *estimator* tracks decrease of V_{oc} based on V_{oc}-SoC relationship and estimated battery drain for every sample time.

V_{oc} *estimator* estimates current V_{oc} using Algorithm 2, and delivers it to other components of *BattTracker*. Algorithm 2 is composed of two main parts: 1) V_{oc} *(re)calibrator*

to (re)calibrate V_{oc} when SoC is reduced by 1% or when V_{oc} is underestimated, i.e., $V_{out} \geq V_{oc}$, and 2) V_{oc} *tracker* to decrease V_{oc} by the amount corresponding to the consumed battery energy.

At the first time *BattTracker* starts, V_{oc} is set to $f(S)$, where S and $f(S)$ are the current SoC and the V_{oc} value obtained from the V_{oc}-SoC look up table, respectively. V_{oc}-SoC look up table for a battery can be obtained based on V_{oc}-SoC curve in Fig. 3.2. Note that V_{oc}-SoC curve is a unique characteristic of a Li-ion battery, which is independent of temperature and aging. Therefore, V_{oc}-SoC curve of a Li-ion battery can be used for other smartphones, employing the same battery product, without any calibration. During run time, V_{oc} *tracker* estimates the current V_{oc} based on instantaneous battery drain rate R_d estimated by R_d *estimator* and SoC obtained by battery interface. The values of V_{oc} between two consecutive SoCs are approximated by linear interpolation of two points. Then, $f(S) - f(S-1)$ is the slope of the line obtained by linear interpolation of $(S-1, f(S-1))$ and $(S, f(S))$, and it denotes average V_{oc} reduction per 1% SoC reduction during $[t^{(S)}, t^{(S-1)}]$. Assuming that estimated R_d during $(t_{k-1}, t_k]$ is $R_d[t_{k-1}]$, the estimated SoC reduction during $(t_{k-1}, t_k]$ can be calculated by $(R_d[t_{k-1}] \cdot t_s/3600)$. V_{oc} reduction corresponding to $(R_d[t_{k-1}] \cdot t_s/3600)$ can be obtained with the slope of the linear interpolation, i.e., $f(S) - f(S-1)$. Accordingly, the amount of estimated V_{oc} reduction at t_k, $\Delta V_{oc}[t_k]$, and V_{oc} estimated at t_k, $V_{oc}[t_k]$, are calculated as follows:

$$\Delta V_{oc}[t_k] = (f(S[t_k]) - f(S[t_k]-1)) \cdot (R_d[t_{k-1}] \cdot t_s/3600), \quad (3.8)$$

$$V_{oc}[t_k] = \max\left(V_{oc}[t_{k-1}] - \Delta V_{oc}[t_k]\,,\; f(S[t_k]-1)\right). \quad (3.9)$$

As shown in (3.9), $V_{oc}[t_k]$ is lower-bounded by $f(S[t_k]-1)$ to avoid underestimation. In summary, V_{oc} is estimated by V_{oc} *estimator* every sample time, and delivered to other components to estimate r_e and R_d.

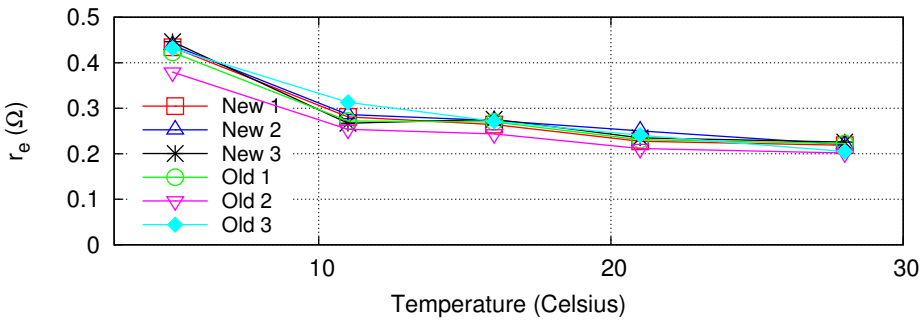


Figure 3.5: Effective resistance r_e of SHV-E210 batteries.

3.5.3 r_e estimator

BattTracker exploits the effective resistance r_e instead of $C(T, d_A)$ and $r(T, d_A)$ to estimate R_d. To enable this, r_e *estimator* estimates r_e during run time and delivers it to R_d *estimator*.

r_e *estimator* exploits (3.5) to estimate r_e. Ideally, cumulative consumed battery energy for $\Delta t^{(n)}$ in Table 3.2 should be 1% of total battery capacity since SoC decreases by 1% for $\Delta t^{(n)}$. Accordingly, r_e is estimated by letting cumulative battery drain for $\Delta t^{(n)}$ be equal to 1% of total battery capacity as follows:

$$\begin{aligned} 1 &= \sum_{t_k \in (t^{(n)}, t^{(n-1)}]} (R_d[t_k] \cdot t_s/3600), \\ &= \sum_{t_k \in (t^{(n)}, t^{(n-1)}]} \frac{100 \cdot (V_{oc}[t_k] - V_{out}[t_k])}{C_f \cdot r_e} \cdot t_s/3600, \end{aligned} \tag{3.10}$$

where the parameters are defined in Table 3.2. Then, r_e is calculated by modifying (3.10) as follows:

$$r_e = \frac{100 \cdot \sum_{t_k \in (t^{(n)}, t^{(n-1)}]} (V_{oc}[t_k] - V_{out}[t_k]) \cdot t_s/3600}{C_f}. \tag{3.11}$$

(3.10) and (3.11) are valid if r_e is constant for $\Delta t^{(n)}$, but r_e varies according to T and d_A as shown in Fig. 3.5. We easily assume that d_A of a battery is constant for $\Delta t^{(n)}$. r_e in Fig. 3.5 varies by $4.1 \pm 3.6\%$ per $1^\circ C$ variation, and hence, r_e can be treated as a constant if temperature does not abruptly changes.

Algorithm 3 r_e *estimator* algorithm

Initializing

1: $A \leftarrow 0, S' \leftarrow SoC$

Waiting for SoC decrease by 1% $S' = SoC$

/*Wait for next sample time*/

2: $S' \leftarrow$ SoC

Estimate r_e **whenever SoC decreases by** 1% BattTracker runs $S' = SoC$

3: Get SoC, V_{out}, and V_{oc}

4: $A \leftarrow A + (V_{oc} - V_{out}) \cdot t_s$

5: $r_e \leftarrow \frac{100 \cdot A}{C_f} \cdot \frac{1}{3600}$, $A \leftarrow 0$, $S' \leftarrow SoC$

The run time algorithm of r_e *estimator* is described in Algorithm 3. If *BattTracker* starts at time t and $S[t]$ is $n\%$, t should be larger than $t^{(n)}$. Therefore, it waits until $t^{(n-1)}$ and starts estimating r_e from $t^{(n-1)}$ (lines 1–1). Then, r_e *estimator* accumulates $(V_{oc} - V_{out}) \cdot t_s$ for every sample time (lines 2–4) and updates r_e whenever SoC changes using (3.11) (line 5). Estimated r_e is delivered to R_d *estimator* so that it estimates R_d without knowing $r(T, d_A)$ and $C(T, d_A)$.

In summary, three components of *BattTracker* interact with each other to estimate variations of V_{oc} and r_e over time and estimate instantaneous R_d by using S and V_{out} which are obtained by battery interface at every sample time.

3.6 Performance Evaluation

BattTracker can be operated on any mobile device powered by the Li-ion battery. For the performance evaluation, we have implemented *BattTracker* with +500 lines of Java codes as an Android background service, and use three SHV-E210 smartphones, one SHV-E300, and one SHV-E330 (Galaxy S4 LTE-A). Two SHV-E210 smartphones run on Android 4.1.2, while the third runs on Android 4.3. Both SHV-E300 and SHV-E330 run on Android 4.4.2. We set t_s to 1 sec for the following evaluations.

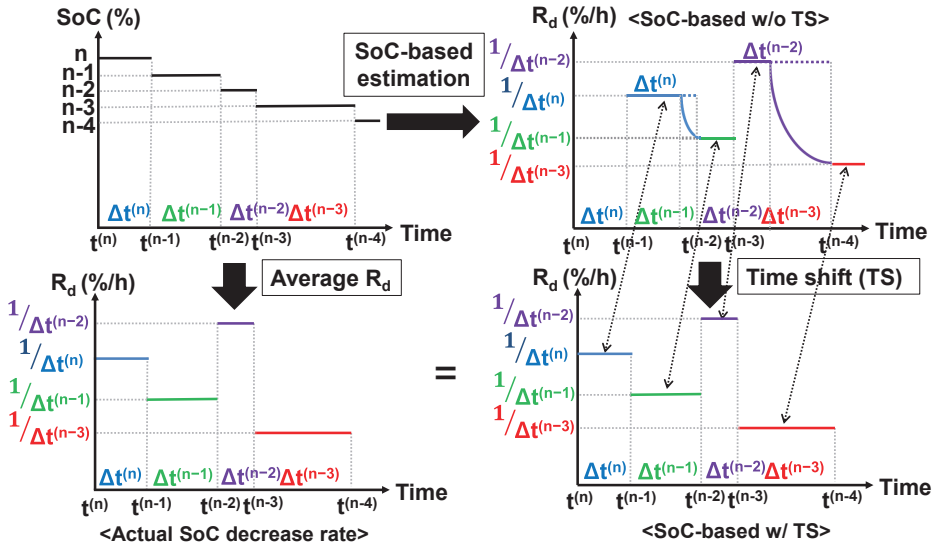


Figure 3.6: SoC variation over time and corresponding SoC decrease rate for $\Delta t^{(n)}$'s (left figures). Estimated R_d by SoC-based w/ and w/o TS (right figures).

3.6.1 Comparison schemes and performance metrics

For comparison schemes, SoC-based scheme is used. SoC-based scheme updates R_d during $\Delta t^{(n+1)}$ at $t^{(n)}$. Fig. 3.6 shows the relationship between SoC variation over time and R_d estimated by SoC-based scheme. For example, it calculates R_d $(= 1/\Delta t^{(n)})$ during $\Delta t^{(n)}$ at $t^{(n-1)}$, and hence, the value updated by SoC-based at $t^{(n-1)}$ is not actual SoC decrease rate during $\Delta t^{(n-1)}$, i.e., $1/\Delta t^{(n-1)}$. We use SoC-based w/ TS which is a time shifted version of SoC-based as a ground truth for $R^{(n)}$ for $\Delta t^{(n)}$, and the average R_d calculated by SoC-based (SoC-based w/o TS) as a comparison scheme. As shown in Fig. 3.6, SoC-based w/o TS revises the estimated R_d when it overestimates R_d, e.g., it estimates $1/\Delta t^{(n)}$ at $t^{(n-1)}$, but it can reduce R_d after $\Delta t^{(n)}$ since it notices the overestimation. Since SoC-based w/ TS cannot be obtained during run time, it is processed after experiment with an off-line method using *BattTracker* logs including S and $t^{(n)}$'s.

For the performance metric, we use mean absolute percentage error (MAPE) of

battery drain rate and cumulative error of estimated SoC decrease (e_{soc}). Since battery drain rate and lifetime are convertible each other, we focus on the battery drain rate. Assuming that we start an experiment at t_s and end at t_e, and SoCs at t_s and t_e are $n_s\%$ and $n_e\%$, respectively. Then, we consider the time duration $(t^{(n_s-1)}, t^{(n_e)}]$ to calculate MAPE and e_{soc}. First, MAPE is defined as the average of the absolute difference between average R_d by *BattTracker* and $R^{(n)}$ by SoC-based w/ TS during $(t^{(n_s-1)}, t^{(n_e)}]$. In other words, we define e_n as the difference between the average R_d by *BattTracker* during $(t^{(n)}, t^{(n-1)}]$ and $R^{(n)}$ by SoC-based w/ TS. Then, MAPE is the average of $|e_n|$ for $n \in \{n_s - 1, n_s - 2, \cdots, n_e + 1\}$. Second, e_{soc} is defined as the ratio of the difference between cumulative battery drain estimated by *BattTracker* (i.e., $\sum R_d[t_k] \cdot \frac{t_s}{3600}$) and actual battery drain (i.e., $n_e - (n_s - 1)$) to the actual battery drain. Therefore, we calculate MAPE and e_{soc} during $(t^{(n_s-1)}, t^{(n_e)}]$ as follows:

$$MAPE = \sum_{n=n_s-1}^{n_e+1} \left(\left| \frac{\sum_{t_k \in (t^{(n)}, t^{(n-1)}]} R_d[t_k])}{(t^{(n-1)} - t^{(n)})} - R^{(n)} \right| / R^{(n)} \right), \tag{3.12}$$

$$e_{soc} = \frac{\left(\sum_{t_k \in (t^{(n_s-1)}, t^{(n_e)}]} \left(R_d[t_k] \cdot \frac{t_s}{3600} \right) \right) - (n_e - (n_s - 1))}{(n_e - (n_s - 1))}. \tag{3.13}$$

3.6.2 Convergence time of r_e

We first evaluate the convergence time of r_e from any initial values of r_e. Fig. 3.7 represents the convergence time of r_e with different initial values when SHV-E210 plays a YouTube video. As shown in Fig. 3.7, r_e converges within the time duration that SoC decreases by 3%. Once r_e is converged to a specific value, the converged r_e can be used as the initial r_e for the next time *BattTracker* restarts, thus reducing the convergence time.

3.6.3 Power consumption overhead of BattTracker

BattTracker provides battery drain rate R_d by accessing system files to obtain V_{out} and S. However, accessing system files via *cat* command causes non-negligible power

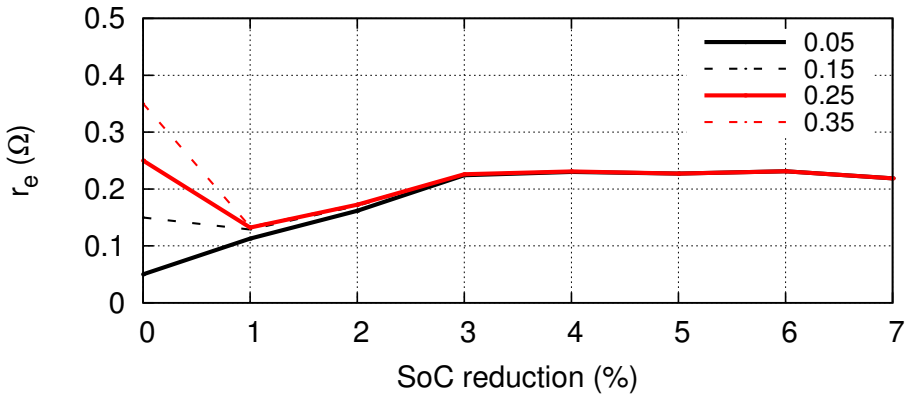
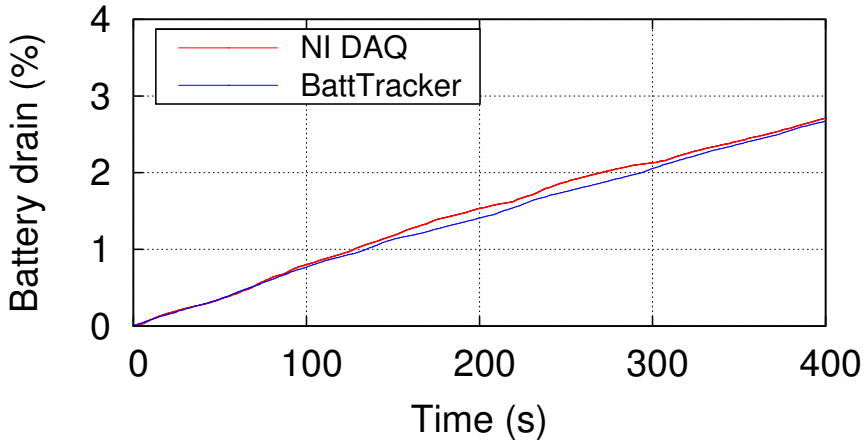


Figure 3.7: Convergence time of r_e with different initial values.

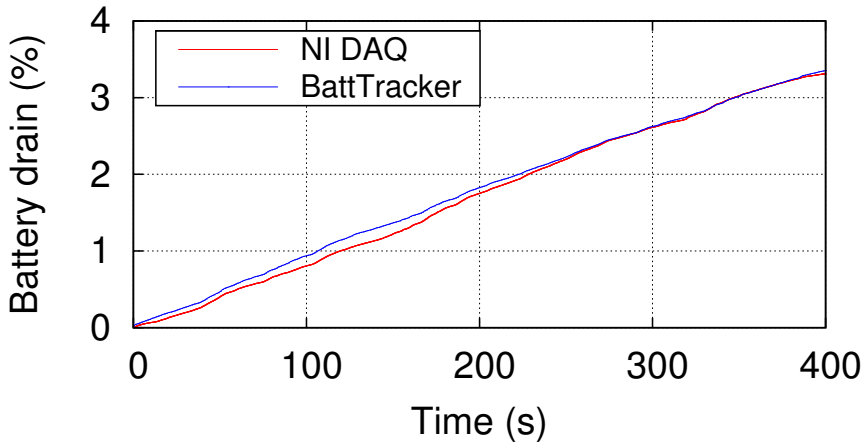
consumption. We evaluate power consumption overhead of *BattTracker* by measuring power consumption of SHV-E300 with and without running *BattTracker* while SHV-E300 smartphone plays a stored video and runs YouTube application with WiFi. Monsoon power monitor is used for measurement. Power consumption overheads for two cases are 51.65 mW and 24.26 mW, respectively, which represent 3.02% and 2.24% of the total power consumption. In idle state, *BattTracker* wakes up CPU for operations, and hence, energy overhead becomes significant. The energy overhead can be optimized by regulating t_s and we leave this as future work. As long as a device is active, we conclude that *BattTracker* does not significantly affect the battery lifetime.

3.6.4 Comparison with measurement using equipment

We evaluate the accuracy of instantaneous R_d estimated by *BattTracker* with measurement result by NI USB-6210 data acquisition (DAQ) [51, 52]. We connect a $50\ m\Omega$ resistor between the positive terminal of the battery and SHV-E300 whereas the negative terminal is directly attached to the device. NI USB-6210 measures the voltage across the resistor to obtain the discharge current (I) of the device while *BattTracker* runs to estimate instantaneous R_d at the same time. Fig. 3.8 shows the cumulative battery drain estimated by *BattTracker* and measured by NI USB-6210 when SHV-E300



(a) YouTube streaming.



(b) Web browsing.

Figure 3.8: Cumulative battery drain measured by NI USB-6210 and estimated by BattTracker for SHV-E300 running (a) YouTube and (b) web browser.

plays (a) YouTube video with WiFi and (b) web surfing. For NI DAQ, discharge current measured by NI DAQ can be translated into R_d by multiplying $\bar{R^{(n)}}/\bar{I}$, where $\bar{R^{(n)}}$ and $\bar{I}$ are average $R^{(n)}$ and I during the measurement time. Cumulative battery drain can be obtained by accumulating $R_d \cdot t_s$ over time. As shown in Fig. 3.8, it is observed that *BattTracker* follows the results of NI DAQ well for both cases.

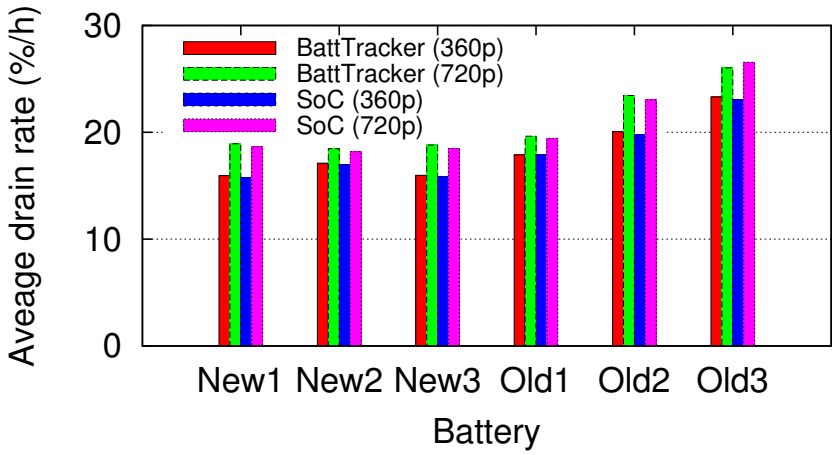
3.6.5 BattTracker with aged batteries

Degree of battery aging affects battery drain rate even if a device consumes the same amount of power since total battery capacity varies according to the degree of aging. We evaluate the performance of *BattTracker* with differently aged batteries of SHV-E210 and SHV-E300 during YouTube video streaming with $360p$ and $720p$ resolutions, respectively. Each video plays for 30 min. Fig. 3.9 shows the average R_d ($E[R_d]$) estimated by *BattTracker* in comparison with that by SoC-based w/ TS during video streaming for various batteries. $E[R_d]$ by SoC-based w/ TS is obtained by $\frac{\{(n_s-1)-(n_e)\}}{(t^{(n_e)}-t^{(n_s-1)})}$ if $S[t_s] = n_s$ and $S[t_e] = n_e$, where t_s and t_e are the start and end time of video, respectively. $E[R_d]$ by *BattTracker* is the average of instantaneous R_d's during $(t^{(n_s-1)}, t^{(n_e)}]$, i.e., $\frac{\sum_{t_k \in (t^{(n_s-1)}, t^{(n_e)}]} R_d[t_k]}{(t^{(n_e)}-t^{(n_s-1)})}$. $E[R_d]$ of aged batteries is larger than that of fresh batteries for both devices, and *BattTracker* accurately estimates $E[R_d]$ regardless of the degree of battery aging effect as shown in Fig. 3.9.

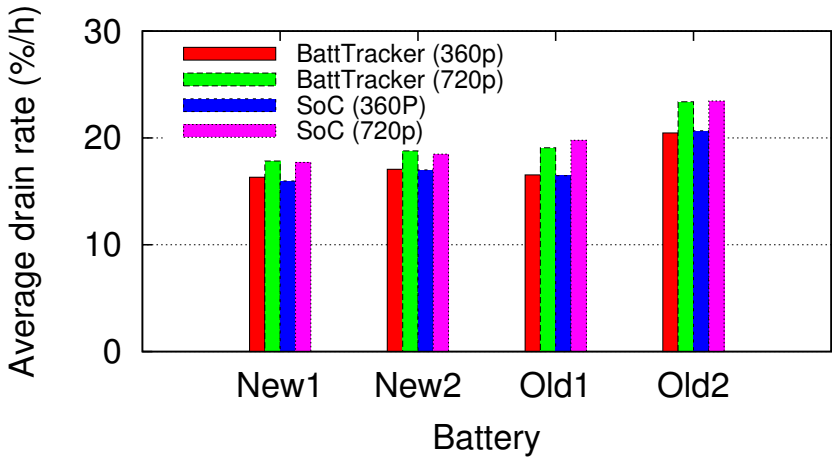
Specifically, we validate the accuracy of *BattTracker* in terms of MAPE. MAPEs (%) of SHV-E210 and SHV-E300 are $1.2 \pm 0.5\%$ and $0.8 \pm 0.4\%$, respectively, and hence, it is validated that *BattTracker* accurately estimates R_d regardless of the degree of battery aging effect.

3.6.6 BattTracker with various video applications

According to the application a device runs, the device consumes different amount of power. For example, a user can watch a video via (a) a streaming application such as YouTube with WiFi/LTE networks, (b) a local video player to play a video in an SD



(a) SHV-E210.



(b) SHV-E300.

Figure 3.9: Battery drain rates for YouTube video streaming with $360p$ and $720p$ resolutions using (a) three fresh batteries ('new') and three aged batteries ('old') of SHV-E210 and (b) two fresh batteries and two aged batteries of SHV-E300.

(secure digital) card, or (c) terrestrial DMB (digital multimedia broadcasting), which is developed based on DAB (digital audio broadcasting, Eureka-147). These three video applications play a video by exploiting different components of the smartphone, and hence, the smartphone consumes different amount of power. For example, the average power of SHV-E210 to play a video via YouTube with WiFi, MX player (a local video player) [53], and DMB are 1.5 W, 0.9 W, and 1.3 W, respectively. Therefore, the battery drain rates should differ each other.

For the comparative performance evaluation of *BattTracker* for various video applications, we play a video via YouTube with WiFi, a stored video with MX player, and DMB, sequentially. Each application runs for 30 min and turn off the screen for 5 min before moving to the next application. We repeat the same experiment with three fresh and three aged batteries for SHV-E210, and two fresh and two aged batteries for SHV-E300. During the measurement time that the smartphones sequentially run three applications, MAPEs of SHV-E210 and SHV-E300 are $1.4 \pm 0.6\%$ and $1.9 \pm 0.8\%$, respectively, and their e_{soc}'s are $-0.31 \pm 2.3\%$ and $-0.57 \pm 2.2\%$, respectively.

Fig. 3.10 partially shows the battery drain rate estimated by *BattTracker*, SoC-based w/, and w/o TS for SHV-E300. Especially, Fig. 3.10 focuses on the transition time from YouTube to MX player. In this figure, BattTracker-MA denotes weighted moving average of R_d ($\tilde{R_d}$) estimated by *BattTracker* as follows:

$$\tilde{R_d}[t_k] = \alpha \cdot \tilde{R_d}[t_{k-1}] + (1 - \alpha) \cdot R_d[t_k], \tag{3.14}$$

where α is 0.9 in this evaluation. BattTracker-$E[R_d]$ represents the average R_d over $\Delta t^{(n)}$ to compare with SoC-based w/ TS. As shown in Fig. 3.10, we stop playing YouTube video and turn off the screen at t_{k1}, and starts MX player to view a video in SD card at t_{k2}. Average $R^{(n)}$ by SoC-based w/ TS are $22.8\%/h$ and $10.81\%/h$ for YouTube and MX player, respectively. During time duration $[t_{k1}, t_{k2}]$, SoC does not change so that we cannot detect the abrupt variation of R_d with SoC-based methods. On the other hand, *BattTracker* can immediately infer the screen off and starting point of new application from variation of the estimated R_d. Furthermore, *BattTracker* es-

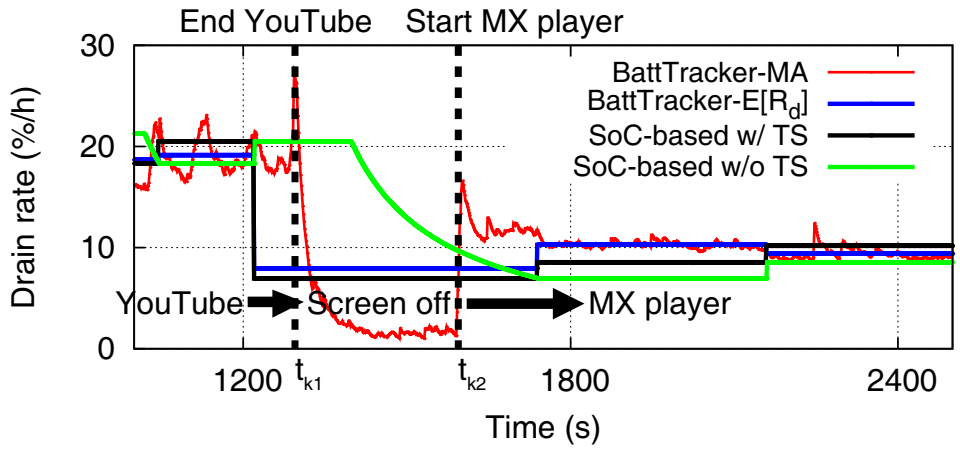


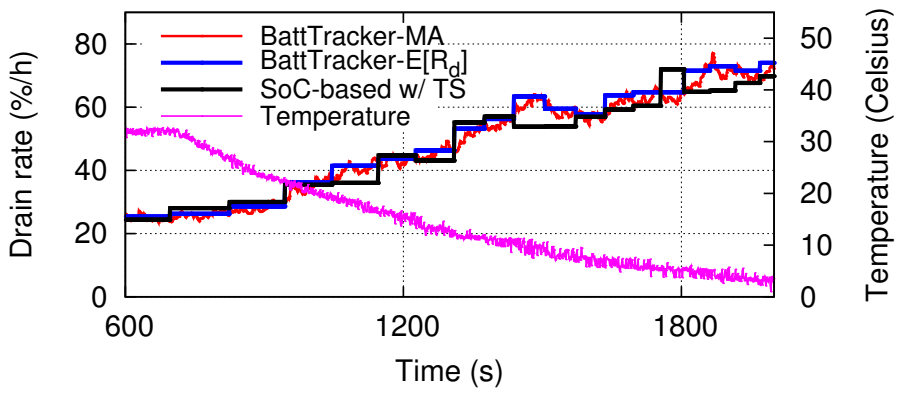
Figure 3.10: R_d estimated by *BattTracker* compared to R_d by SoC-based w/ TS and w/o TS. SHV-E300 plays a YouTube video with WiFi and plays a stored video after 5 min screen off.

timates average R_d of a certain application within a few seconds, whereas SoC-based needs more than a few minutes.

3.6.7 BattTracker with varying temperature

Even though a device consumes the same amount of power, battery drain rate varies according to temperature since total battery capacity varies according to the temperature. We evaluate the performance of *BattTracker* at various temperatures with fresh and aged batteries. We put three smartphones in a refrigerator to vary temperatures while the smartphones play a YouTube video with WiFi. The power consumption of three smartphones for this case is measured as $1.2-1.5$ W with less fluctuation in long-term time scale.

Fig. 3.11 shows the estimated R_d's by *BattTracker* and SoC-based w/ TS for SHV-E210 with an aged battery. BattTracker-MA and BattTracker-$E[R_d]$ increase from about 20% to 70% as the temperature decreases from $35\ ^\circ C$ to $0\ ^\circ C$, and this trend is well-matched with that of SoC-based w/ TS. MAPEs of SHV-E210, SHV-E300, and SHV-E330 with fresh and aged batteries are $4.3 \pm 0.5\%$, $3.1 \pm 0.3\%$, and $3.5 \pm 0.4\%$,



Estimated R_d by *BattTracker* compared to SoC-based w/ TS when SHV-E210 plays a YouTube video with WiFi in the refrigerator.

respectively, and their e_{soc}'s are $-2.1 \pm 0.2\%$, $-4.5 \pm 0.1\%$, and $-4.6 \pm 0.5\%$, respectively. Accordingly, we conclude that *BattTracker* estimates the battery drain rate accurately irrespective of both temperature and battery aging.

3.7 Summary

In this chapter, we propose *BattTracker*, which estimates instantaneous battery drain rate and remaining lifetime for mobile devices. *BattTracker* considers the effects of temperature and battery aging on battery characteristics such as capacity and internal resistance by using the effective resistance (r_e) concept. It accurately estimates battery drain rate and lifetime irrespective of temperature and battery aging by efficiently tracking V_{oc} and r_e. *BattTracker* can achieve up to 0.5 second time granularity, thus enabling us to realize energy-awareness for mobile applications and provide fine-grained battery drain rate for users. Our extensive evaluation demonstrates that *BattTracker* accurately provides the battery drain rate of any aged battery at any temperature.

Chapter 4

REQUEST: Seamless Dynamic Adaptive Streaming over HTTP for Multi-Homed Smartphone under Resource Constraints

4.1 Introduction

Multi-homing enables smartphones to utilize multiple network interfaces, e.g., LTE and Wi-Fi, simultaneously to enhance the quality of experience (QoE) for the end users. Among various mobile applications, dynamic adaptive streaming over HTTP (DASH) is a very good candidate that can take advantage of multi-homing towards better QoE [28, 29]. Mobile users can enjoy high-quality video over Wi-Fi in high-bandwidth Wi-Fi hotstpots, and watch video contents over LTE whenever LTE base stations can support data communication even though there are no available Wi-Fi access points (APs). In a place where both LTE and Wi-Fi are available, higher quality videos can be supported or more resource efficient streaming becomes possible by judicially using both LTE and Wi-Fi links simultaneously.

Let us consider a video streaming scenario, where user wants to enjoy content with long playback duration, e.g., a live soccer game or video on demand (VoD) contents from YouTube or Netflix. Since the content duration is very long, Wi-Fi may be the de-

sirable access network for the user. However, the user might be on move or take public transport while experiencing the content. Such mobility results in extremely fluctuating link throughput or frequently disconnected Wi-Fi links from the nearby hotspots [8,9]. This significantly degrades video quality and causes frequent rebuffering (also called video stall or freezing), and hence, in this environment, it is not desirable to watch video over Wi-Fi link.

To tackle this problem, an intelligent video chunk requesting mechanism should be developed that can utilize the unstable Wi-Fi links intelligently while guaranteeing the desired video quality and uninterrupted playback without rebuffering in mobile environments. At the same time, battery energy and LTE data quota are important issues which are desired to be conserved as much as possible while ensuring satisfying video quality.

Accordingly, a chunk request technique should consider both enhancing video performances, i.e., video quality and rebuffering, and saving smartphone resources, i.e., battery energy and LTE data quota, to optimize QoE of video streaming. Besides, the problem is more complex in a multi-homing environment compared with a single link scenario [24], thus making it challenging to develop a well-designed DASH streaming client, and a solution is still due.

To provide an optimal video performance while satisfying resource usage constraints for battery energy and LTE data quota, we propose REQUEST, a bit*R*ate, *E*nergy, LTE data *Q*uota, and b*U*ff*E*r-aware video *ST*reaming, for DASH video over multi-homed smartphones. By utilizing Wi-Fi link as a supplementary link, REQUEST realizes a seamless DASH video streaming even in the environments where Wi-Fi link performance is not guaranteed. Also, REQUEST optimizes time-average video quality while satisfying time-average resource constraints by adopting Lypunov optimization framework, and it is easily implemented in commercial smartphones as an application. We claim the following major contributions:

- We propose a chunk request policy that achieves seamless playback of video using

both Wi-Fi and LTE simultaneously even in situations where Wi-Fi is unstable.

- We formulate a Lyapunov optimization framework-based stochastic optimization problem to maximize time-average video quality under time-average energy and LTE data usage constraints and minimize rebuffering.
- We design REQUEST, an online video chunk request algorithm by using both LTE and Wi-Fi links, which provides a near-optimal solution of the Lyapunov optimization problem.
- We implement REQUEST by modifying ExoPlayer, Google's open-source DASH player for Android [34], and validate its performance in real-world scenarios.

The rest of the chapter is organized as follows. In Section 4.2, we discuss issues arising when utilizing both LTE and Wi-Fi for DASH video streaming and related work. Section 4.3 describes motivation of our work, and our proposed chunk request policy is presented in Section 4.4. We formulate the optimization problem in Section 4.5 and REQUEST algorithm is introduced in Section 4.6. We evaluate the performance of REQUEST in Section 4.7 and conclude the chapter in Section 4.8.

4.2 Background and Related Work

4.2.1 Background

Multi-homing for DASH

DASH is a bitrate-adaptive streaming technique, which delivers video content over HTTP. Video content is encoded at a variety of bitrates in video server. A DASH-enabled video streaming player downloads video chunks of a particular bitrate based on the experienced bandwidth for downloading earlier chunks. Initially, the player begins with lower or moderate quality to avoid longer start-up delay. Thanks to the development of techniques for simultaneous utilization of multiple network interfaces

at mobile devices, especially, as application-level solutions [54, 55], both Wi-Fi and LTE interfaces can be utilized simultaneously to enhance video quality while saving battery energy and LTE data quota. By downloading video chunks with both LTE and Wi-Fi, video player can maintain sufficient chunks in its buffer, thus avoiding rebuffering. However, chunks remaining in the buffer may cause possible data waste when user stops watching video in the middle of playback [24, 26].

Bandwidth aggregation for enhancing video quality: Traditional bandwidth aggregation in heterogeneous wireless network environment aims to support higher bitrates of video that cannot be sufficiently delivered by only one of the available networks [56–60].

Wi-Fi offloading to reduce LTE data usage: User may not fully utilize LTE link due to her/his remaining monthly LTE data quota or data plan. In this case, it is fairly useful to download video data by Wi-Fi as much as possible to save LTE data quota [8, 61].

Energy consumption: The higher the video quality, the higher the energy consumption of smartphones, because the amount of data to be decoded and downloaded via Wi-Fi or LTE increases [29, 43]. The energy is also spent by the CPU for processing more packets [43].

Fast prefetching: Prefetching video data, i.e., requesting more video chunks in advance, provides several advantages to video client. The more video chunks to be consecutively requested, the less energy is spent for networking activities as network interfaces are able to stay in idle state for longer periods [10,24,26]. In addition, prefetching prevents rebuffering events because video buffer is filled with sufficiently many video chunks. However, large amount of prefetching may waste as much energy and LTE data as the number of chunks remaining in the video buffer when user stops watching video before the end of the video clip. The ratio of wasted energy and LTE data by prefetching large amount of chunks becomes significant, especially when user abandons video streaming session quite early. Therefore, it is necessary to determine an optimal prefetching strategy considering the advantages of prefetching (prevention of

rebuffering and energy efficiency) and disadvantages (waste of energy and LTE data due to user's leaving) to enhance QoE of video consumers.

Lyapunov optimization framework

Stochastic optimization aims at minimizing a time-average objective function subject to queue stability when the utility function and queue stability conditions are in tradeoff relationship. In addition, the stochastic optimization framework is able to utilize the concept of virtual queues for time-average constraint representation. In stochastic optimization formulation, Lyapunov function $L(t)$ is defined as the sum of squares of backlogs in actual and virtual queues on a slot (in a slotted system). Lyapunov drift $\Delta(t)$ is defined as the difference in the Lyapunov function per slot time. While pursuing the minimization of a time-average objective function, taking the minimum of the Lyapunov drift leads to the queue stability (i.e., main constraint), which is referred to as *drift-plus-penalty* minimization. By taking an action to greedily minimize the drift-plus-penalty every slot time, we can achieve a time-average utility deviating by at most $O(1/V)$ from optimality while satisfying time-average constraints and a time-average queue backlog bound of $O(V)$, where V is defined as a tradeoff factor between utility and queue stability. For further details about the theory of Lyapunov optimization, the book [62] can be referred to.

4.2.2 Related Work

Energy/cellular data quota-aware video streaming: Previous efforts [23,24] control a video segment size to be prefetched in an HTTP-based video streaming service to improve the energy efficiency. *GreenTube* [23] aims to minimize an unnecessary active period of 3G/4G radios by scheduling each video segment downloading. *eSchedule* [24] utilizes crowd-sourced video viewing statistics and power models for energy efficient scheduling of video streaming. *QAVA* [25] manages the tradeoff between cellular data usage and video quality by predicting video client's usage behavior. *QAVA*

automatically selects optimal video quality to enable users to keep under their data quota while maximizing video quality. Lee *et al.* [26] and Hu *et al.* [27] consider wastage of energy and cellular data quota incurring when user stops watching video. Both consider the scenario where video streaming is conducted only via LTE interface of a smartphone. *GreenBag* [28] utilizes both LTE and Wi-Fi links to achieve better quality of service (QoS) and energy efficiency. Go *et al.* [29] propose an energy-aware HTTP adaptive video streaming under a cellular data usage constraint. It considers simultaneous usage of LTE and Wi-Fi for DASH video streaming on smartphones. It selects video bitrate and the number of chunks to request via each network in order to minimize a weighted sum of video distortion and energy consumption per video chunk.

Optimal bitrate selection of DASH video: Video client selects a video chunk with an appropriate bitrate according to current network status [30, 31] or video player's buffer status [32]. Due to severe fluctuation of link throughput in mobile environment, it is difficult for a link throughput-based bitrate adaptation to practically function, and for this reason, a buffer-based bitrate adaptation has been studied and practically used by real implementations [32, 33].

4.3 Motivation

4.3.1 Wi-Fi Throughput Fluctuation

While a mobile device can utilize both Wi-Fi and LTE networks simultaneously, the Wi-Fi link may become unstable and its quality may fluctuate severely especially in a mobile or dense environment. For example, when a user moves around and goes out of the coverage of a Wi-Fi AP connected to the user's mobile device, the device cannot utilize Wi-Fi link until it finds an available Wi-Fi AP nearby and handover to a new Wi-Fi AP is successfully completed. It is also well known that Wi-Fi throughput degradation occurs in mobile environments due to poor performance of handover operations

in commercial Wi-Fi devices [3–5]. In addition, if a Wi-Fi AP operates at 2.4 GHz, the Wi-Fi link quality may severely suffer from interference caused by other wireless devices operating in 2.4 GHz ISM band, such as Bluetooth, ZigBee, microwave ovens, and cordless phones [6, 7]. Furthermore, Wi-Fi at 5 GHz suffers from higher path loss than Wi-Fi at 2.4 GHz, thus increasing the possibility that mobile user experiences worse Wi-Fi link quality and goes out of Wi-Fi coverage. In contrast to Wi-Fi, a device may retain a seamless connection via the LTE network even though user moves fast, thanks to seamless handover between LTE base stations.[1] Likewise, in mobile and dense environments, Wi-Fi link may have more unstable quality and sometimes may be unavailable.

However, Wi-Fi can be still utilized for offloading purpose even in mobile environment [8, 9], and its offloading capability may increase in static environment or when a device associates to an IEEE 802.11ac-compliant Wi-Fi AP that can provide very high throughput. From this perspective, when a DASH client requests video chunks via both Wi-Fi and LTE in parallel in a multi-homed device, Wi-Fi link availability is like a double-edged sword. In other words, although the Wi-Fi link may enhance the video quality at low energy cost and reduce LTE data consumption, it can also increase the possibility of rebuffering events, as we cannot predict the exact bandwidth and availability of Wi-Fi in mobile and dense environments.

Therefore, it is challenging to design a chunk request policy for DASH by judicially utilizing Wi-Fi connectivity in addition to LTE to maximize the merit of utilizing Wi-Fi link while minimizing its side effects. In this work, we accept the challenge. We opportunistically request video chunks over Wi-Fi, thus reducing the side effects of unstable Wi-Fi links.

[1] In this chapter, it is assumed that seamless handover of LTE links is ensured. Other cellular links, e.g., 3G/HSDPA, may be applied instead of LTE.

Table 4.1: Important notations

Symbol	Description
t_p	A fixed chunk playback time (sec)
T_b	Initial buffer-time (sec)
r	The index of request event
$t[r]$	Start time of the rth request event
$T[r]$	The rth request interval ($t[r+1] - t[r]$)
$b[r]$	Bitrate of chunks to be requested in the rth request event
$N_l[r]$	Number of chunks to request over LTE in the rth request event
$N_w[r]$	Number of chunks to request over Wi-Fi in the rth request event
$\hat{N}_w[r]$	Number of chunks actually received over Wi-Fi in the rth request event

4.3.2 Optimizing Resource Utilization

Mobile devices consume resources, i.e., battery energy and LTE data quota, during DASH video streaming over Wi-Fi and LTE networks. Since battery energy and remaining LTE data quota are usually limited, users will like to minimize the resource usage for DASH video streaming as much as possible so as to watch videos much longer or use the remaining resources for other applications. Assume that a DASH client intelligently requests proper bitrate video chunks to balance the quality of video and resource usage. In this case, the video quality might be either increased in exchange of using more resources or decreased to conserve the resources. From this perspective, a problem can be formulated as optimizing video quality for DASH streaming given the constrained amount of resources in smartphones, i.e., requesting high quality video chunks with a given amount of battery energy and LTE data quota.

Unfortunately, the traditional optimization frameworks that have been used in the past studies do not support flexible resource utilization, e.g., sometimes allowing re-

source overuse to enhance video quality. For instance, a popular method to formulate optimization problem is to formulate a multi-attribute cost function with proper constraints based on a simple additive weighting (SAW) method. This approach is widely used for multi-attribute decision making (MADM) algorithms [29, 63, 64]. A challenge to optimize MADM-based cost function is to select appropriate weights for the attributes in the cost function to quantitatively balance them. In addition, even though the weights of the attributes determine relative priority to provide a trade-off between attributes in a cost function, it is difficult to force an attribute to have a value within a certain range.

To overcome the limitation of MADM-based optimization, we adopt Lyapunov optimization framework to optimize time-average video quality while satisfying time-average resource constraints at the same time. By adopting time-average concepts, we occasionally allow resource overuse but eventually satisfy constraints.

4.4 Proposed Chunk Request Policy

Generally, before a DASH client sends a request for a video chunk, it determines an appropriate bitrate for the chunk according to the estimated link bandwidth [29] or the playback buffer status [32, 33]. In a multi-homed environment, a DASH client can further download a single chunk in parallel via multiple wireless interfaces by sending multiple HTTP range requests [65]. A number of proposed approaches follow this technique [28, 29]. However, considering a possible situation where Wi-Fi link is disconnected or extremely unstable during a chunk download, it is not efficient to divide one video chunk into two parts and receive them by both LTE and Wi-Fi separately. Accordingly, in this work, we use a single network to request and receive a single video chunk. This eliminates decoding failures due to partially received chunks, thus avoiding re-buffering and data wastage.

Therefore, we design our chunk request policy as illustrated in Fig. 4.1, where the

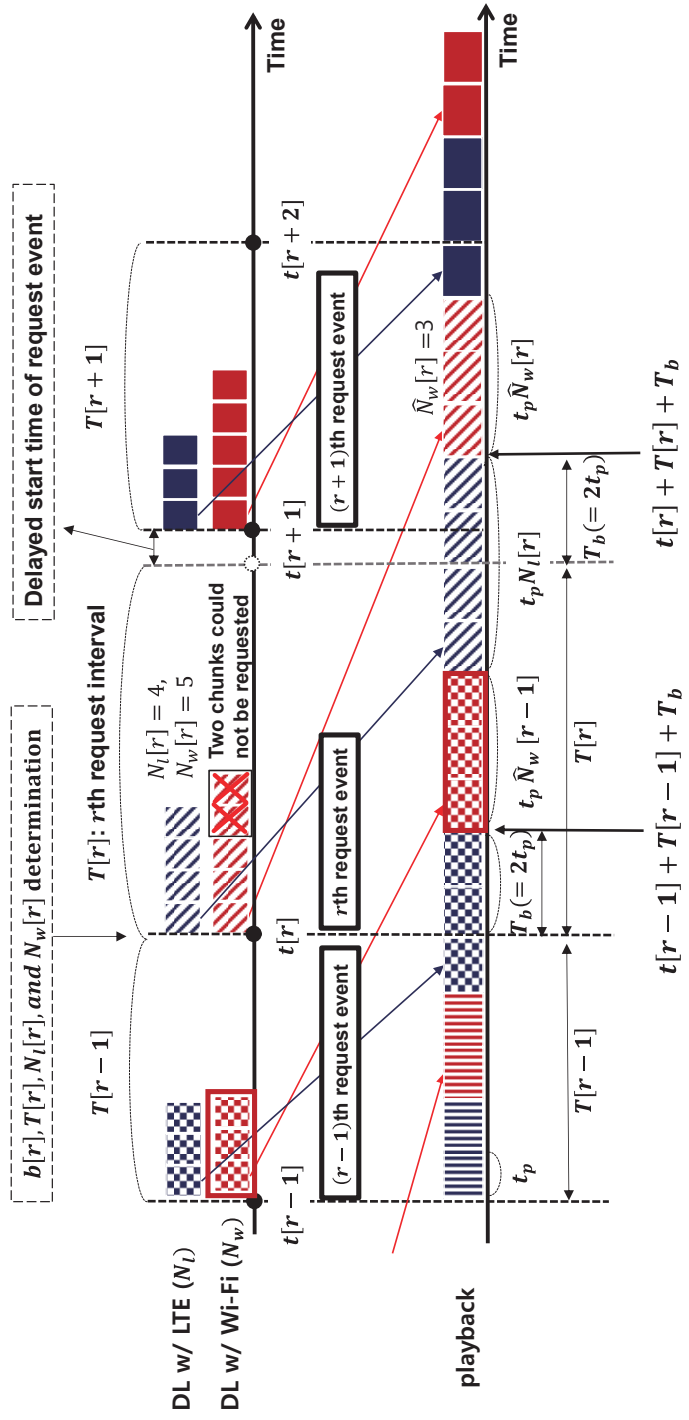


Figure 4.1: An example of the proposed chunk request policy.

notations in Fig. 4.1 are summarized in Table 4.1. We call an event of requesting a batch of video chunks a *request event*. A *request interval* $T[r]$ is defined as the time interval between the start time and the end time of a *request event*. At the start time $t[r]$ of the rth *request event*, client determines the bitrate $b[r]$ of chunks,[2] *request interval* $T[r]$, and the numbers of chunks to request over LTE ($N_l[r]$) and Wi-Fi ($N_w[r]$) during $T[r]$. Ideally, all the chunks requested during $T[r]$, i.e., $N_l[r] + N_w[r]$ chunks, are expected to be successfully downloaded within $T[r]$. In this case, $t[r+1] = t[r]+T[r]$, i.e., the end time of the rth *request event* is equal to the start time of the $(r + 1)$th *request event*.

Moreover, it is important to determine $T[r]$ judiciously to ensure that all the chunks are downloaded before they are played back for seamless video streaming. Especially, our policy tries to avoid rebuffering due to Wi-Fi throughput degradation. To enable this goal, the following features are introduced in our design:

i) We first set $T[r] = t_p(\hat{N}_w[r-1]+N_l[r])$, i.e., the playback time of $\hat{N}_w[r-1]$ chunks received over Wi-Fi in the $(r - 1)$th *request event* and $N_l[r]$ chunks requested over LTE in the rth *request event*. By doing so, the chunks received over Wi-Fi accumulate in the buffer, which has the same effect as increasing the initial buffer-time at the start of the next *request event*. It also prevents rebuffering due to LTE throughput degradation that cannot be handled by the initial buffer.

ii) The video chunks requested over Wi-Fi in the rth *request event* should be played after T_b from the end time of the rth *request event*, where T_b is initial buffer-time.[3] For example, if $T_b = 2t_p$, $\hat{N}_w[r]$ chunks received over Wi-Fi in the rth *request event* are played after $t[r] + T[r] + 2t_p$ as illustrated in Fig. 4.1.

iii) If chunks are requested over both LTE and Wi-Fi, the video chunk requested over Wi-Fi is played right after the last video chunk requested over LTE in the same

[2] All the chunks requested in a *request event* have the same bitrate.

[3] The initial buffer-time is the playback time of video chunks initially received before video player starts rendering the video.

request event. In other words, the first sequence of $N_w[r]$ chunks is the right next to the last sequence of $N_l[r]$.

iv) If client fails to request some chunks over Wi-Fi, those chunks are prioritized to request in the next *request event*. For example, as illustrated in Fig. 4.1, if two chunks could not be requested in the rth *request event*, the client starts requesting the chunks in order in the $(r+1)$th *request event* for the continuity of video.

Based on these features, we ensure that all chunks of a video can be eventually received and played even though some chunks could not be requested over Wi-Fi during a *request event*. However, in practice, chunks may not be received within a *request interval*, due to throughput degradation of either LTE or Wi-Fi. In this case, the start of the next *request event* can be delayed. The details are discussed in Section 4.6.1. In addition, to maximize video quality while satisfying battery energy and LTE data quota constraints, we should properly determine b, T, N_l, and N_w for each *request event*. To achieve this, we formulate a stochastic optimization problem in the following section.

4.5 Problem Formulation

Even if Wi-Fi and LTE links are sufficient to support the highest video quality, we cannot request high quality video if the battery energy and/or LTE data usage is limited. Thus, we have to accurately determine the bitrate of chunks and the number of requested chunks to maximize video quality while satisfying battery and LTE data usage constraints during the entire video playback.

Objective: The objective of our optimization problem is to maximize a time-average video utility. In this work, we define the time-average utility as a time-average logarithmic function of the video bitrate. The reason why we maximize the logarithmic function of bitrate instead of bitrate directly is to reflect the fact that the impact of increasing video quality on a user experience can be modeled by a concave function with diminishing returns [33].

Constraints: To guarantee longer battery lifetime and use LTE data without exceeding budget, we assume that mobile device is allowed to consume p_{av} (W) and consume LTE data with d_{av} (Mbps) on average during a video playback.

In order to design optimal chunk request policy for maximizing the time-average video quality with satisfying time-average resource constraints, it would require *a priori* statistical knowledge, such as the distribution of network bandwidth. It would also need a complex computational method, such as dynamic programming (DP) method for finding the optimal solution based on the *a priori* statistical knowledge [66]. However, it is practically difficult to obtain the exact distribution of network bandwidth to solve the problem with a DP method. Even if the distribution could be obtained, very large state space composed of request time, bitrate, number of chunks to request via network would have to be constructed [33, 67].

In order to overcome this challenge, we apply Lyapunov optimization-based dynamic algorithm [67] that independently determines the number of chunks to request and their bitrate at the start of a *request event* by observing the chunk reception result of the previous *request event*. The performance of Lyapunov-based dynamic algorithm is close to that of the optimal solution by a DP algorithm with *a priori* statistical knowledge, but Lyapunov optimization algorithm does not require any *a priori* knowledge [67].

Renewal-based Lyapunov optimization: According to the chunk request policy proposed in Section 4.4, *request interval* $T[r]$ is related to the number of requested chunks, and hence, it is time-varying. To reflect this, we adopt renewal-based Lyapunov optimization [62, 67].[4] We assume that the video playback time is infinite, i.e., $R \to \infty$,[5] for an approach similar to that in [33]. Then, the problem based on Lyapunov optimization framework to maximize time-average video utility while satisfying time-average

[4] A *request event* in Fig. 4.1 corresponds to a renewal frame in the renewal-based Lyapunov optimization.

[5] R is the index of the last *request event* for a video.

energy and LTE data usage constraints is described as follows:

$$\begin{aligned} &\text{Maximize} \quad \overline{\log(b)T}/\overline{T}, \\ &\text{subject to} \quad \overline{e}_e \leq p_{av}\overline{T}, \;\; \overline{e}_d \leq d_{av}\overline{T}, \end{aligned} \tag{4.1}$$

where $\overline{\log(b)T}, \overline{T}, \overline{e}_e$, and $\overline{e}_d$ are infinite horizon expectations, i.e.,

$$(\overline{\log(b)T}, \overline{T}, \overline{e}_e, \overline{e}_d) = \lim_{R\to\infty} \frac{1}{R} \sum_{r=0}^{R-1} (\log(b[r])T[r], T[r], e_e[r], e_d[r])), \tag{4.2}$$

where $e_e[r]$ and $e_d[r]$ are the energy consumption and LTE data usage during the rth *request event*, respectively. Since *request interval* is time-varying, we give a time weight to $\log(b)$, thus maximizing $\overline{\log(b)T}/\overline{T}$, which denotes the time-average video utility.

Virtual queues for resource constraints: We define virtual queues to solve our optimization problem based on a drift-plus-penalty ratio minimization algorithm, which greedily minimizes drift-plus-penalty ratio to achieve near-optimal time-average video utility while satisfying time-average resource constraints.

Let us first define a virtual queue for energy consumption $Q_e[r]$. The arrival and service process of $Q_e[r]$ at the rth *request event* are $e_e[r]$ and $p_{av}T[r]$, respectively. Then, the queue backlog of $Q_e[r]$ evolves as follows:

$$Q_e[r+1] = \max\left(Q_e[r] + e_e[r] - p_{av}T[r], 0\right). \tag{4.3}$$

If i) $Q_e[0] = 0$, ii) $Q_e[r]$ satisfies (4.3) for all $r \in \{0, 1, 2, \cdots\}$, and iii) $Q_e[r]$ is mean-rate stable (i.e., $lim_{R\to\infty} \frac{Q_e[R]}{R} = 0$) [62], then $\overline{e}_e \leq p_{av}\overline{T}$ for all integers $R > 0$.

Lemma 4.5.1. *If $Q_e[0] = 0$ and $Q_e[r]$ satisfies (4.3) for all $r \in \{0, 1, 2, \cdots\}$ and $Q_e[r]$ is mean rate stability [62], then for all integers $R > 0$:*

$$\overline{e}_e \leq p_{av}\overline{T} \tag{4.4}$$

Proof. From (4.3) we have:

$$\begin{aligned} &Q_e[r+1] \geq Q_e[r] + e_e[r] - p_{av}T[r], \\ &Q_e[r+1] - Q_e[r] \geq e_e[r] - p_{av}T[r]. \end{aligned} \tag{4.5}$$

Summing over $r \in \{0, 1, \cdots, R-1\}$ provides:

$$Q_e[R] - Q_e[0] \geq \sum_{r=0}^{R-1} e_e[r] - p_{av} \sum_{r=0}^{R-1} T[r] \tag{4.6}$$

Diving by R, using $Q_e[0] = 0$, and taking the limit $R \to \infty$, we obtain $\lim_{R\to\infty} \frac{Q_e[R]}{R} \geq \overline{e}_e - p_{av}\overline{T}$. Since $Q_e[r]$ is mean rate stability, i.e, $\lim_{R\to\infty} \frac{Q_e[R]}{R} = 0$, and hence, we obtain

$$\overline{e}_e[r] \leq p_{av}\overline{T}. \tag{4.7}$$

□

In addition, from (4.6), for all $R > 0$ such that $Q_e[R] = 0$, the average energy consumption $\frac{1}{R}\sum_{r=0}^{R-1} e_e[r]$ is less than or equal to $\frac{1}{R}p_{av}\sum_{r=0}^{R-1} T[r]$, i.e., the inequality constraint of (4.1) is satisfied.

The same approach can be applied to the LTE data usage virtual queue $Q_d[r]$, where the queue backlog evolution of $Q_d[r]$ is represented as follows:

$$Q_d[r+1] = \max\left(Q_d[r] + e_d[r] - d_{av}T[r], 0\right). \tag{4.8}$$

Drift-plus-penalty ratio: The renewal-based Lyapunov optimization framework minimizes drift-plus-penalty ratio which is obtained by normalizing drift-plus-penalty by time [67]. Before defining the drift-plus-penalty ratio, we first define the Lyapunov function $L[r]$. In addition to Q_e and Q_d, we consider DASH client's video buffer $Q_r[r]$ which denotes the number of video chunks in video buffer at the rth *request event*. Then, $L[r]$ is defined by:

$$L[r] = \frac{1}{2}\left(\left(Q_e[r]\right)^2 + \left(Q_d[r]\right)^2 + \left(\frac{Q_{\max}}{2} - Q_r[r]\right)^2\right), \tag{4.9}$$

where $Q_{\max}$ is defined as the maximum number of chunks stored in the video buffer. The energy consumption and LTE data usage queues are desired to be empty to satisfy the time-average constraints while the video chunk queue should not be empty to avoid rebuffering. However, filling the video buffer to Q_{max} is not desirable due to the

possibility of queue overflow and waste of resources, and hence, we use $\frac{Q_{max}}{2}$ instead of Q_{max} when defining $L[r]$.

Then, the Lyapunov drift $\Delta[r]$, which is desired to be minimized, is defined as the difference between the Lyapunov function of the rth *request event* and that of the $(r+1)$th *request event*.

$$\Delta[r] = L[r+1] - L[r]. \tag{4.10}$$

Conventional Lyapunov optimization framework minimizes penalty. Since our work maximizes video utility, we define penalty as $-\log(b)T$ to follow the Lyapunov framework's concept of penalty minimization. Finally, the drift-plus-penalty ratio (DPP_r) is defined as:

$$DPP_r[r] = \frac{\Delta[r] + V \cdot penalty[r]}{T[r]} = \frac{\Delta[r]}{T[r]} - V\log(b[r]), \tag{4.11}$$

where V is a positive constant parameter to allow a tradeoff between virtual queues' backlogs and the gap from a theoretical optimal video utility. By taking an action to greedily minimize the drift-plus-penalty ratio every *request event*, we can achieve time-average video utility maximization deviating by at most $O(1/V)$ from optimality while satisfying time-average resource constraint bound of $O(V)$.

4.6 REQUEST Algorithm

In this section, we introduce REQUEST, an online algorithm to request video chunks with near-optimal video quality while satisfying resource constraints and preventing video rebuffering. The REQUEST architecture is illustrated as Fig. 4.2. At the start of every *request event*, REQUEST observes current virtual queue backlogs ($Q_e[r]$ and $Q_d[r]$), video queue backlog ($Q_r[r]$), and estimated LTE and Wi-Fi link throughput ($\tilde{r}_l[r]$ and $\tilde{r}_w[r]$). By using these values, REQUEST determines the four parameters, i.e., video bitrate ($b[r]$), *request interval* ($T[r]$), and the number of chunks to request through LTE and Wi-Fi ($N_l[r]$ and $N_w[r]$), which minimize the drift-plus-penalty ratio,

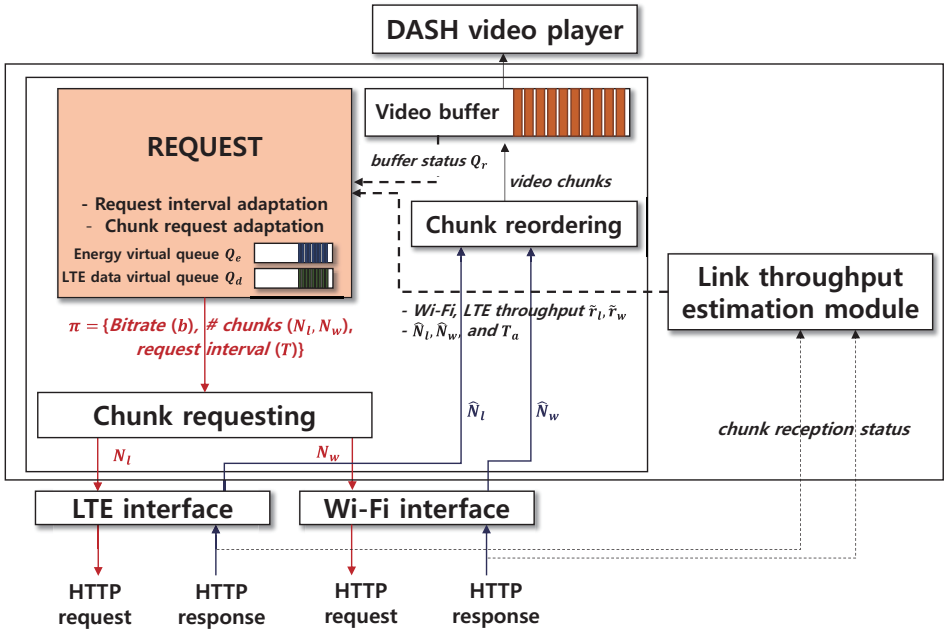


Figure 4.2: REQUEST architecture.

$DPP_r[r]$, in (4.11). Let the four-tuple $(N_l[r], N_w[r], b[r], T[r])$ be the *request policy* $\pi[r]$ for the rth *request event*.

The main algorithms of REQUEST include *request interval adaptation* and *chunk request adaptation*. The *request interval adaptation* controls *request interval* according to chunk reception status, and the *chunk request adaptation* determines π based on DPP_r minimization algorithm.

4.6.1 Request Interval Adaptation

Delayed start time of request event: If all the chunks requested in the rth *request event* are successfully downloaded within $T[r]$, $t[r+1]$ is equal to $t[r]+T[r]$. However, in practice, some chunks requested via either LTE or Wi-Fi may not be successfully received on time, i.e., within $T[r]$, due to severe throughput fluctuation. In such a case, we should delay the start time of the next *request event* for successful chunk

receptions. If we request $N_l[r]$ and $N_w[r]$ chunks by LTE and Wi-Fi at the rth *request event*, respectively, the delay is allowed until the following conditions are satisfied.

i) We wait until all the chunks requested over LTE ($N_l[r]$) are successfully received.

ii) If a chunk is requested over Wi-Fi before $t[r] + T[r]$, we wait until $t[r] + T[r]$ for full reception of the chunk.

Let the number of actually received chunks during the rth *request event* by LTE and Wi-Fi be $\hat{N}_l[r]$ and $\hat{N}_w[r]$. With the above two conditions, $\hat{N}_l[r] = N_l[r]$ and $\hat{N}_w[r] \leq N_w[r]$. The reason why we do not guarantee the reception of all the chunks requested over Wi-Fi is that we quickly start a new *request event* and adapt the bitrate and the number of chunks for the next *request event* to resolve link throughput degradation.

If the chunk download of the rth *request event* is not completed within $T[r]$ and it takes additional $T_a[r]$ time to finish the remaining chunk reception, the start time of the $(r+1)$th *request event* $t[r+1]$ becomes $t[r]+T[r]+T_a[r]$. Fig. 4.3 shows the examples for the start time of the next *request event* when chunks are requested with $N_l = 4$ and $N_w = 5$ for the rth *request event*. Fig. 4.3(a) shows the case that all the requested chunks are received within $T[r]$. In Fig. 4.3(b), download of the last chunk (the fourth chunk) requested over LTE does not finish until $t[r] + T[r]$, and $t[r + 1]$ is delayed by $T_a[r]$. In Fig. 4.3(c), both of the last chunks requested over LTE and Wi-Fi have not been received until $t[r] + T[r]$, and the download over LTE finishes earlier than the one over Wi-Fi so that $T_a[r]$ is determined by the download completion time of the fourth Wi-Fi chunk. As shown in Fig. 4.3(d), even though there are two chunks left to request over Wi-Fi, those chunks will not be requested over Wi-Fi after $t[r] + T[r]$.

Request interval adaptation: When a delay of T_a occurs, the delay is compensated by reducing the next *request interval* by T_a. If $\hat{N}_w[r - 1]$ chunks are received over Wi-Fi and the delay of $T_a[r - 1]$ occurs, $T[r]$ is reduced by $T_a[r - 1]$ as follows:

$$T[r] = t_p(\hat{N}_w[r - 1] + N_l[r]) - T_a[r - 1]. \tag{4.12}$$

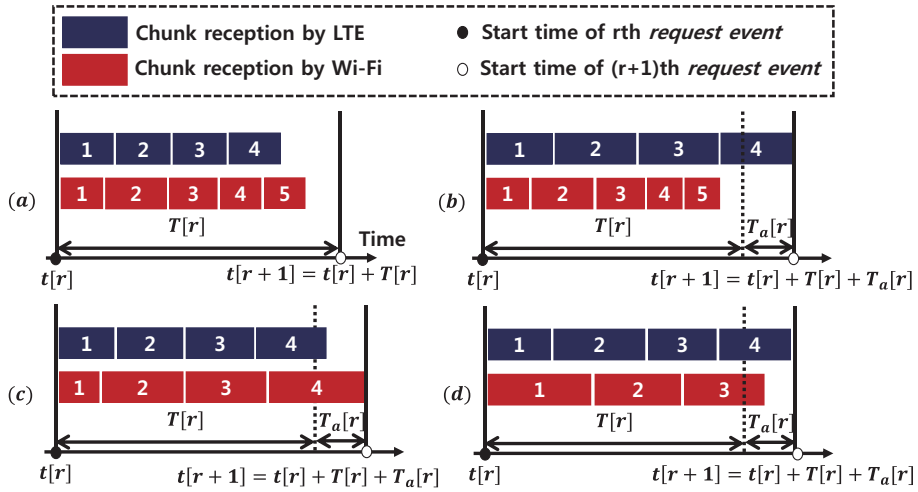


Figure 4.3: Examples for request interval and start time of the next request event according to the chunk download results.

By doing so, we can compensate for the amount of buffer that has been reduced due to the delay. For example, even though the rth *request event* is delayed by $T_a[r-1]$, we can ensure that there are $\hat{N}_w[r]+T_b/t_p$ chunks in the buffer at $t[r+1]$ by reducing $T[r]$ by $T_a[r-1]$. Without this adaptation, the delays accumulate and initial buffer-time may not be guaranteed at the start time of a *request event*, thus causing rebuffering. According to our request interval adaptation, the video buffer backlog evolves as:

$$Q_r[r+1] = Q_r[r] + (\hat{N}_w[r] + N_l[r]) - (T[r] + T_a[r])/t_p. \tag{4.13}$$

Lemma 4.6.1. *If $Q_r[1] = N_b$, $T[r]$ is determined by (4.12), and $Q_r[r]$ evolves as (4.13) for all $r \in \{1, 2, 3, \cdots\}$, then for all integers $R > 0$, there is no video stall if $T_a[R] < N_b \cdot t_p$.*

Proof. From (4.12) and (4.13) we have:

$$\begin{aligned} Q_r[r+1] - Q_r[r] =& (\hat{N}_w[r] + \hat{N}_l[r]) - (T[r] + T_a[r])/t_p, \\ Q_r[r+1] - Q_r[r] =& (\hat{N}_w[r] + \hat{N}_l[r]) - (\hat{N}_w[r-1] + N_l[r]) \\ & + (T_a[r] - T_a[r-1])/t_p, \\ Q_r[r+1] - Q_r[r] =& \hat{N}_w[r] - \hat{N}_w[r-1] + (T_a[r] - T_a[r-1])/t_p. \end{aligned} \tag{4.14}$$

Summing over $r \in \{0, 1, \cdots, R-1\}$ provides:

$$\begin{aligned} Q_r[R] - Q_r[1] =& \hat{N}_w[R-1] - (T_a[R-1] - T_a[0])/t_p, \\ Q_r[R] =& N_b + \hat{N}_w[R-1] - (T_a[R-1])/t_p, \\ Q_r[R] \geq& N_b - T_a[R-1]/t_p > 0. \end{aligned} \tag{4.15}$$

□

4.6.2 Chunk Request Adaptation

Estimated LTE and Wi-Fi throughput: For the rth request event, the estimated LTE/Wi-Fi throughput, $\tilde{r}_l[r]$ and $\tilde{r}_w[r]$, are calculated by using exponential weighted moving average (EWMA).

$$\tilde{r}_l[r] = \alpha \tilde{r}_l[r-1] + (1-\alpha) r_l[r-1], \tag{4.16}$$

where $r_l[r-1] = \frac{N_l[r-1] t_p b[r-1]}{\tau_l[r-1]}$, i.e., the average chunk download speed by LTE during the $(r-1)$th *request event*, and $\tau_l[r-1]$ is the total download time for $N_l[r-1]$ chunks of bitrate $b[r-1]$ by the LTE interface. $\tilde{r}_w[r]$ is calculated in a similar fashion.

Arrival rate of energy/LTE data virtual queues: In order to reduce energy and LTE data waste when a user stops watching video in the early stage of playback, we put the expected energy/LTE data waste as well as the expected energy/LTE data consumption in the arrival process of the energy/LTE data virtual queue in the stage of determining the optimal $\pi[r]$. We define $\bar{E}_w[r]$ and $\bar{D}_w[r]$ as the expected energy and LTE data

waste during the rth *request event*, respectively.[6]

The arrival rate $e_e[r]$ of energy virtual queue is given as

$$e_e[r] = e_v[r] + e_{e,w}[r] + e_{e,l}[r] + \bar{E}_w[r] \cdot T[r]/(t[r] + T[r]), \tag{4.17}$$

where $e_v[r]$ is the energy consumption for video playback including CPU and display power, $e_{e,w}[r]$ and $e_{e,l}[r]$ are the energy consumption for Wi-Fi and LTE interfaces, respectively. We multiply the expected waste by $\frac{T[r]}{t[r]+T[r]}$ to reflect the effect of waste amount on the average power from the beginning of video playback until the end of the current *request event*. The expected energy consumption for network interface n_i, where $n_1 = l$ and $n_2 = w$ for LTE and Wi-Fi, respectively, is calculated as follows:

$$e_{e,n_i}[r] = \tilde{P}_{n_i}[r] t_{n_i,d}[r] + P_{n_i,tail} \cdot \min\left((T[r] - t_{n_i,d}[r]), t_{n_i,tail}\right), \tag{4.18}$$

where $\tilde{P}_{n_i}[r]$ is the average power for network n_i with the expected throughput $\tilde{r}_{n_i}[r]$, and $t_{n_i,d}[r]$ $\left(= \frac{t_p b[r] N_{n_i}[r]}{\tilde{r}_{n_i}[r]}\right)$ is the expected download time for $N_{n_i}[r]$ chunks of bitrate $b[r]$ by network n_i with the expected throughput $\tilde{r}_{n_i}[r]$. $\tilde{P}_{n_i}[r] t_{n_i,d}[r]$ is the energy consumption of network interface n_i for chunk download and the remaining term denotes the energy consumption for tail time [43].

Likewise, the arrival rate $e_d[r]$ of LTE data virtual queue is

$$e_d[r] = t_p b[r] N_l[r] + \bar{D}_w[r] \cdot T[r]/(t[r] + T[r]). \tag{4.19}$$

For $\bar{E}_w[r]$ and $\bar{D}_w[r]$, we assume that the probability that a user stops watching video during $T[r]$ takes a uniform distribution. During $T[r]$, the video chunk remaining in the buffer, which are received via Wi-Fi in the previous *request event*, will be played back for $t_p Q_r[r]$, and the chunk received by LTE in the current *request event* will be played back for the remaining time, i.e., $T[r] - t_p Q_r[r]$. We adopt the approach in [26] to derive E_w and D_w as the following closed forms.

[6]When updating $Q_e[r]$ and $Q_d[r]$ at $t[r]$, since the video has been played back continuously, set $\bar{E}_w[r-1]$ and $\bar{D}_w[r-1]$ to 0, and then update $Q_e[r]$ and $Q_d[r]$.

$$\begin{aligned}\bar{E}_w[r] =& \frac{1}{T[r]}\left\{\frac{1}{2}(t_p Q_r[r])^2 e_{e,w}[r-1] + \left(T[r] - \frac{1}{2}t_{l,d}\right) e_{e,l}[r] \right. \\ &\left. + \left(T[r] - \frac{1}{2}t_{w,d}\right) e_{e,w}[r] + \frac{(T[r] - t_p Q_r[r])^2}{2t_p N_l[r]} e_{e,l}[r]\right\}.\end{aligned} \tag{4.20}$$

$$\bar{D}_w[r] = \left(1 - \frac{t_{l,d} + t_p N_l[r]}{2T[r]}\right) e_d[r]. \tag{4.21}$$

Chunk request policy determination algorithm: Now, we find the optimal $\pi[r]$ which minimizes $DPP_r[r]$ as follows:

$$\begin{aligned}&\min_{\pi[r]} \ DPP_r[r], \\ &\text{subject to } \ T[r] = t_p\left(\hat{N}_w[r-1] + N_l[r]\right) - T_a[r-1], \\ &\qquad \frac{t_p b[r] N_l[r]}{T[r]} \le \tilde{r}_l[r], \frac{t_p b[r] N_w[r]}{T[r]} \le \tilde{r}_w[r], \\ &\qquad N_l[r], N_w[r] \in \{0, 1, 2 \cdots, Q_{max}\}, \\ &\qquad 1 \le N_l[r] + N_w[r] \le Q_{max}.\end{aligned} \tag{4.22}$$

The first constraint of (4.22) is to follow the chunk request policy in Section 4.4. The second constraint is to prevent requesting more chunks than what the estimated link throughput permits. The third and fourth constraints represent that the video should be requested in a unit of chunk. The run time algorithm to obtain the optimal $\pi[r]$ is summarized in Algorithm 4. At the start of a new *request event*, update Q_e, Q_d, and Q_r, and estimate $\tilde{r}_l$ and $\tilde{r}_w$ (line 3). For all the possible $\pi[r]$ (lines 3–4), check whether π satisfies the second constraint, and if π is feasible, calculate DPP_r (lines 4–5). The π minimizing DPP_r is selected as the optimal request policy (line 6).

4.7 Performance Evaluation

Comparison scheme: Although many schemes for video streaming have been proposed to improve QoE, there have been few studies to consider DASH-based video streaming by using both LTE and Wi-Fi simultaneously. We select the energy-efficient

Algorithm 4 Optimal chunk request policy determination

Initialize:

1: $Q_e \leftarrow 0$, $Q_d \leftarrow 0$, $Q_r \leftarrow T_b/t_p$, and $N_{w,prev} \leftarrow 0$

During video playback: Video plays New *request event starts*

2: $N_{w,prev} \leftarrow \hat{N}_w$, $DPP_{min} \leftarrow \infty$

3: Update Q_e, Q_d, Q_r, Estimate l$\tilde{r}_l$ and $\tilde{r}_w$

$N_t \in \{1, 2, \cdots, Q_{max}\}$ $N_l \in \{1, 2, \cdots, N_t\}$

4: $N_w \leftarrow N_t - N_l$, $T \leftarrow t_p(N_{w,prev} + N_l) - T_a$ $b \in \{b_1, b_2, \cdots, b_m\}$ $\pi = \{N_l, N_w, T, b\}$ is feasible

5: Calculate e_e, e_d, L, Δ, and DPP_r $DPP_r \leq DPP_{min}$

6: $DPP_{min} \leftarrow DPP_r$, $\pi^{opt} \leftarrow \pi$

7: Request video chunks according to π^{opt}

8: Wait for the next start time of *request event*

HTTP adaptive streaming algorithm (EE-HAS) proposed in [29] as a comparison scheme. To our best knowledge, EE-HAS is the only scheme that can be implemented as an application without modifying other protocol stacks. EE-HAS considers energy consumption and LTE data usage constraint to determine the optimal bitrate of video chunks. The major difference between REQUEST and EE-HAS is that EE-HAS divides each video chunk into two segments, which are requested over LTE and Wi-Fi in parallel.

Performance metrics: the following metrics are used in this work.

i) *Average video quality:* We measure the average video bitrate during the entire playback time. For REQUEST, the average bitrate is calculated as $\frac{\sum_{r=1}^{R} b[r](\hat{N}_l[r]+\hat{N}_w[r])}{\sum_{r=1}^{R}(\hat{N}_l[r]+\hat{N}_w[r])}$, where R is the index of the last *request event*.

ii) *Rebuffering:* Rebuffering occurs if a video buffer becomes empty or some chunks have not been successfully received within the time limit due to abrupt throughput degradation or Wi-Fi disconnection in the middle of a chunk download. Frequent rebuffering events severely degrade QoE of user [33, 68, 69].

iii) *Amount of energy and LTE data waste:* If a user stops watching a video at time instant t, let the waste of energy and LTE data at t be $E_w(t)$ and $D_w(t)$, respectively. We analyze the effect of energy and LTE data waste on overall power and LTE data usage rate, respectively, by dividing $E_w(t)$ and $D_w(t)$ by t. We refer to $E_w(t)/t$ and $D_w(t)/t$ as the time-normalized energy waste and LTE data waste, respectively.

Parameters: In both simulation and measurement, we use $\alpha = 0.5$ for (4.16), $Q_{max} = 10$, and $t_p = 4$ (s). Video starts after downloading two initial chunks, i.e., $T_b = 8\ s$. For the measurement, we use 596-second Big-Buck-Bunny video clip[7] encoded at 20 bitrates, i.e., $\{$0.045, 0.089, 0.128, 0.177, 0.218, 0.256, 0.323, 0.378, 0.509, 0.578, 0.783, 1.009, 1.207, 1.474, 2.087, 2.410, 2.944, 3.341, 3.614, 3.936$\}$ (Mbps). For the simulation, each trial runs for 600 s, and uses the same video encoded at 10 bitrates, i.e., $\{$0.23, 0.33, 0.48, 0.69, 0.99, 1.43, 2.06, 2.96, 5.03, 6.00$\}$ (Mbps).[8]

4.7.1 Prototype-Based Evaluation

We implemented both REQUEST and EE-HAS by modifying the open-source DASH Android application, ExoPlayer. We use Samsung Galaxy S5 smartphone (SM-G900) which runs on Android 5.0. For both schemes, the same power model of SHV-E120 smartphone [43] is used to estimate the energy consumption for chunk download and video playback. [9] We conduct an experiment in a lab environment, where average LTE throughput ranges from 10 to 15 Mbps. We set the maximum throughput of Wi-Fi to 10 Mbps by controlling the QoS option in the setting window of Wi-Fi AP. In the middle of video playback, we degrade Wi-Fi throughput to less than 2 Mbps by adding a contending node with saturated UDP uplink traffic for 180 s.

[7] http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/BigBuckBunny/4sec/.

[8] For simulation, we used video encoded with higher bitrate than measurement case since LTE and Wi-Fi throughput measured in real environments was much higher than the highest bitrate used in the measurement.

[9] Our work is independent of power models, i.e., any power model can be used for this REQUEST algorithm.

REQUEST with various resource constraints

We first evaluate REQUEST with various power, p_{av} (W), and LTE data, d_{av} (Mbps), constraints, i.e., $\{(p_{av}, d_{av}) | p_{av} \in \{1.5, 2, 3\}, d_{av} \in \{1, 2, 3, 4\}\}$. Fig. 4.4 shows the average bitrate, power, and LTE data usage rate during the entire playback. During the period when there is no Wi-Fi background traffic from other connected devices, REQUEST fully utilizes Wi-Fi link to save LTE data usage while satisfying the power constraint, and hence, the average LTE data usage is generally less than the LTE data constraint. Obviously, REQUEST can achieve almost the highest bitrate with the highest power and LTE data constraints. Besides, with tight resource constraints, REQUEST tends to select lower average bitrate to satisfy the resource constraints. For all the cases, REQUEST satisfies the constraints.

REQUEST without resource waste consideration

REQUEST considers the expected energy and LTE data waste when calculating the arrival rate of energy and LTE data virtual queues using (4.3) and (4.8). To evaluate the resource saving by considering the expected waste, we implemented another version of REQUEST, namely REQUEST-WO, which does not consider the expected waste. Fig. 4.5 shows the expected waste, i.e., $E_w(t)$ and $D_w(t)$, and normalized waste, i.e., $E_w(t)/t$ and $D_w(t)/t$, when p_{av}=2 W and d_{av}=1 Mbps. For EE-HAS, $\alpha = 0.3$. Since EE-HAS is prefetching aggressively, it is more wasteful than REQUEST in the early stage of video playback. Before the video plays for 50 s, both $E_w(t)/t$ and $D_w(t)/t$ of REQUEST are smaller than half of those of REQUEST-WO. REQUEST reduces the waste by selecting lower bitrate and smaller number of chunks to request compared to REQUEST-WO at the beginning of the video. Since the resource waste is relatively high compared to the total resource usage if user stops watching video in a few seconds, REQUEST can save more resources when user quits video early. REQUEST can be a more resource-saving option, and enabling REQUEST-WO can be also an alternative if user simply wants to watch higher video quality with more potential waste.

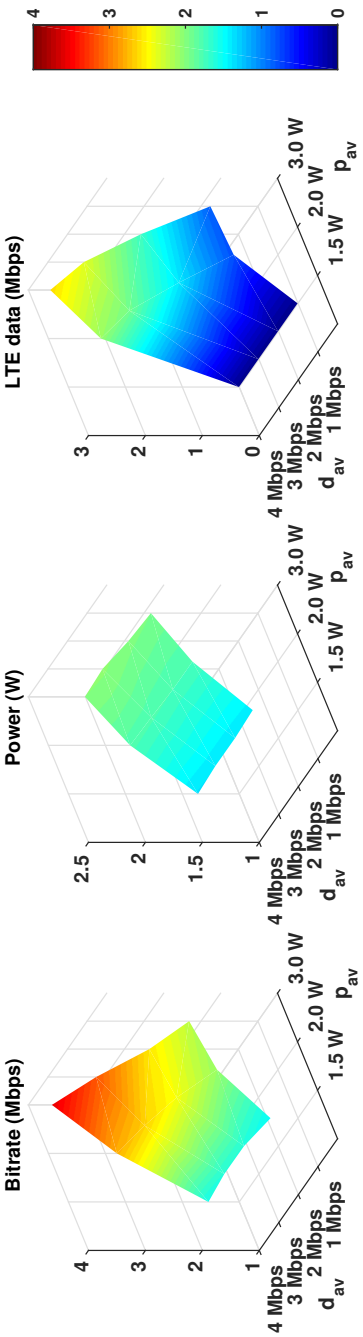
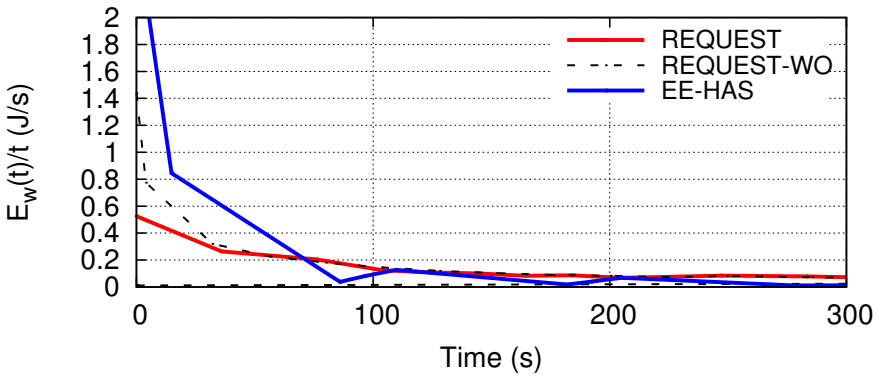
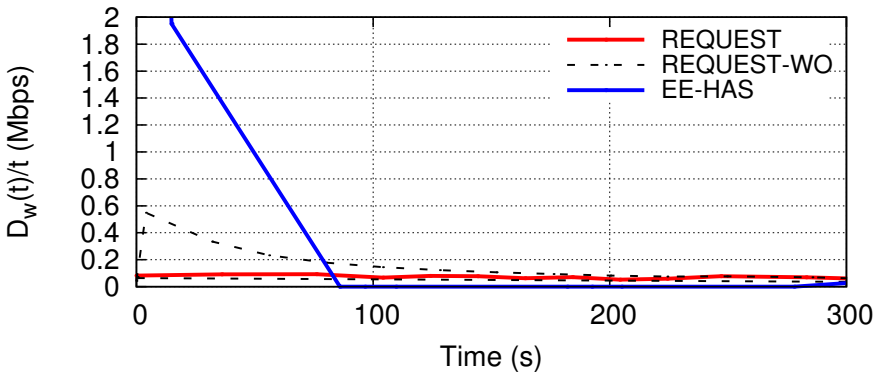


Figure 4.4: Average bitrate, power, LTE data usage rate of REQUEST. The x-label and y-label of each colormap are power and LTE data constraints, respectively.



(a) Time-normalized energy waste.



(b) Time-normalized LTE data waste.

Figure 4.5: Time-normalized energy and LTE data waste comparison for REQUEST and REQUEST-WO.

Comparison study

For comparative evaluation of REQUEST, we run REQUEST and EE-HAS with various p_{av} and α. Fig. 4.6 presents the average bitrate and total rebuffering time during the entire playback time of the 596-second video. Playback smoothness ($\%$) is defined as $\frac{100 \cdot \text{Total playback time}}{\text{Total playback time + rebuffering time}}$. EE-HAS with small α increases bitrate but suffers from long total rebuffering time, i.e., 41.0 s for $\alpha = 0.3$, and total 59.6 s for $\alpha = 0.1$ when the background traffic starts to degrade Wi-Fi throughput abruptly. Even though EE-HAS with large α, which operates more energy efficiently, does not suffer rebuffering, the average bitrate is quite low, i.e., below 100 kbps with $\alpha = 0.9$. However, REQUEST retains 100% playback smoothness for all the power constraints with over 1 Mbps average bitrate. In addition, the average bitrates of EE-HAS with $\alpha = 0.4$ and REQUEST with $p_{av} = 1.5$ are similar, i.e., over 1.5 Mbps, but EE-HAS shows only 93.6% playback smoothness (40.5 s rebuffering time) while REQUEST provides seamless playback. In summary, in contrast to EE-HAS which provides either too low bitrate without rebuffering or high bitrate with long rebuffering time, REQUEST achieves enhanced quality without rebuffering.

4.7.2 Trace-Driven Simulation

To comparatively evaluate the performance of REQUEST, we implement both REQUEST and EE-HAS using Matlab. For a fair comparison, measured LTE and Wi-Fi TCP throughput traces are used. We first measured the LTE and Wi-Fi TCP throughput using *Iperf* with SM-G900 every second in various places, e.g., office, subway station, and cafeteria. A desktop in a lab is used as a server for *Iperf*.

We use $\alpha = 0.1$ for EE-HAS and $p_{av} = 2\ W$ for REQUEST. d_{av} is 1.5 Mbps. We classify rebuffering events into two types, i.e., i) rebuffering due to empty video buffer (empty buffer rebuffering) and ii) rebuffering due to absence of video chunks to be played now even though video buffer is filled with video chunks to be played later (out-of-order rebuffering). The out-of-order rebuffering may occur especially with EE-

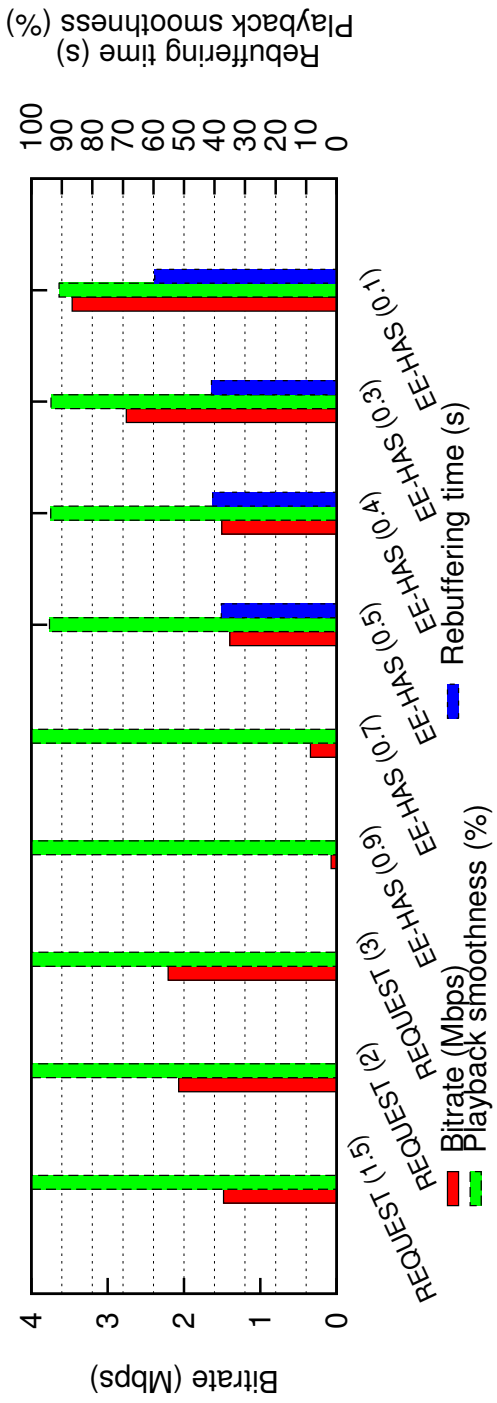


Figure 4.6: Average bitrate, playback smoothness, and rebuffering time of REQUEST (p_{av}) and EE-HAS (α) with various p_{av} and α. For all the cases, $d_{av} = 1\ Mbps$.

Table 4.2: Bitrate and rebuffering of EE-HAS ($\alpha = 0.1$) and REQUEST ($p_{av} = 2$) for five traces.

	EE-HAS						REQUEST	
Trace	RT (s)	# ER	ER time (s)	# OR	OR time (s)	Bitrate (Mbps)	RT (s)	Bitrate (Mbps)
1 (office)	4	1	4	0	0	5.27	0	5.17
2 (station)	16.39	3	5.33±1.89	1	0.4	3.03	0.30	5.14
3 (station2)	43.83	3	8±3.27	3	6.61±3.81	2.00	0	2.22
4 (cafe)	12	3	4	0	0	4.06	0	5.17
5 (cafe2)	32.65	3	6.67±1.89	16	0.79±0.16	2.59	0	5.23

HAS, where a chunk may be divided into two segments that are requested over LTE and Wi-Fi, separately in parallel. If these segments are not received by the player on time, a partially received chunk cannot be decoded and played back, thus causing out-of-order rebuffering. In Table 4.2, # ER, and ER time denote the number of empty buffer rebuffering events and the average buffering time per event. # OR and OR time are those of the out-of-order rebuffering. RT denotes the total rebuffering time of REQUEST, and bitrate (Mbps) is the average bitrate during playback. For all the cases, rebuffering time of REQUEST is almost zero while EE-HAS suffers from frequent rebuffering events, e.g., 19 rebuffering events and total 32.65 s rebuffering time for the third trace. Bitrate of REQUEST is similar to or higher than that of EE-HAS while REQUEST avoids rebuffering even in mobile and dense environments where LTE and Wi-Fi throughput are often unstable.

4.8 Summary

In this chapter, we propose REQUEST which utilizes both LTE and Wi-Fi networks to provide seamless video streaming under resource constraints. The proposed chunk request policy of REQUEST ensures that all the video chunks are received even in unstable network environments. REQUEST achieves near-optimal time-average video qual-

ity while satisfying time-average resource constraints by adopting Lyapunov optimization framework. Our prototype-based evaluation and trace-driven simulation demonstrate that REQUEST provides seamless video streaming by using both LTE and Wi-Fi links in real-world environments.

Chapter 5

Concluding Remarks

5.1 Research Contributions

In this dissertation, we have addressed energy-aware technologies for smartphone which utilizes both LTE and Wi-Fi network interfaces.

In Chapter 2, we have presented PIMM, a power modeling for smartphone which utilizes both LTE and Wi-Fi network interfaces. By decomposing packet processing power and network interface power, PIMM accurately predicts the estimated energy consumption when a smartphone utilizes both LTE and Wi-Fi network interfaces. Through measurement with various smartphones and applications, we validate that our model outperforms other existing power models for single/multiple network transmissions in terms of the estimation error.

In Chapter 3, we have proposed *BattTracker*, which estimates instantaneous battery drain rate for smartphones. *BattTracker* considers the effects of temperature and battery aging on battery characteristics such as capacity and internal resistance by using the effective resistance (r_e) concept. It accurately estimates battery drain rate irrespective of temperature and battery aging. *BattTracker* can achieve up to 0.5 second time granularity, thus enabling us to realize energy-awareness for mobile applications and provide fine-grained battery drain rate for users. Our extensive evaluation demon-

strates that *BattTracker* accurately provides the battery drain rate of any aged battery at any temperature.

In Chapter 4, we have proposed REQUEST which utilizes both LTE and Wi-Fi networks to provide seamless video streaming under resource constraints. The proposed chunk request policy of REQUEST ensures that all the video chunks are received even in unstable network environments. REQUEST achieves near-optimal time-average video quality while satisfying time-average resource constraints by adopting Lyapunov optimization framework. Our prototype-based evaluation and trace-driven simulation demonstrate that REQUEST provides seamless video streaming by using both LTE and Wi-Fi links in real-world environments.

5.2 Future Work

As further improvement on the results of this dissertation, there are several research items as follows.

First, regarding power modeling of multiple network interface-activated smartphones, our future work will extend PIMM to cover multiple spatial streams and channel bonding in IEEE 802.11n and 802.11ac.

Second, regarding battery drain rate estimation, we will extend *BattTracker* to estimate the battery drain rate when battery is charged by either USB or AC charger.

Lastly, regarding DASH video streaming, our future work will use both PIMM and *BattTracker* to estimate and update energy consumption for video chunk download, respectively. We will also develop a run time adaptation of the control parameter V of Lyapunov optimization for *REQUEST* to better control the trade-off between video quality and resource utilization.

Bibliography

[1] J. Vetter, P. Novak, M. Wagner, C. Veit, K.-C. Möller, J. Besenhard, M. Winter, M. Wohlfahrt-Mehrens, C. Vogler, and A. Hammouche, "Ageing mechanisms in lithium-ion batteries," *Elsevier Journal of power sources*, vol. 147, no. 1, pp. 269–281, 2005.

[2] S. Lee, J. Kim, J. Lee, and B. Cho, "State-of-charge and capacity estimation of lithium-ion battery using a new open-circuit voltage versus state-of-charge," *Elsevier Journal of Power Sources*, vol. 185, no. 2, pp. 1367–1373, 2008.

[3] V. Mhatre and K. Papagiannaki, "Using smart triggers for improved user performance in 802.11 wireless networks," in *Proc. ACM MobiSys*, 2006.

[4] I. Ramani and S. Savage, "SyncScan: practical fast handoff for 802.11 infrastructure networks," in *Proc. IEEE INFOCOM*, 2005.

[5] J. Shi, L. Meng, A. Striegel, C. Qiao, D. Koutsonikolas, and G. Challen, "A walk on the client side: monitoring enterprise WiFi networks using smartphone channel scans," in *Proc. IEEE INFOCOM*, 2016.

[6] K.-H. Kim, H. Nam, and H. Schulzrinne, "WiSlow: A Wi-Fi network performance troubleshooting tool for end users," in *Proc. IEEE INFOCOM*, 2014.

[7] S. Rayanchu, A. Patro, and S. Banerjee, "Airshark: detecting non-WiFi RF devices using commodity WiFi hardware," in *Proc. ACM IMC*, 2011.

[8] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can WiFi deliver?" *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 2, pp. 536–550, 2013.

[9] S. Bae, D. Ban, D. Han, J. Kim, K. H. Lee, S. Lim, W. Park, and C. K. Kim, "StreetSense: effect of bus Wi-Fi APs on pedestrian smartphone," in *Proc. ACM IMC*, 2015.

[10] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. ACM MobiSys*, 2012, pp. 225–238.

[11] A. Garcia-Saavedra, P. Serrano, A. Banchs, and G. Bianchi, "Energy consumption anatomy of 802.11 devices and its implication on modeling and design," in *Proc. ACM CoNEXT*, 2012, pp. 169–180.

[12] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang, "Fine-grained power modeling for smartphones using system call tracing," in *Proc. ACM EuroSys*, 2011, pp. 153–168.

[13] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. IEEE/ACM/IFIP CODES+ISSS*, 2010, pp. 105–114.

[14] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proc. ACM MobiCom*, 2012, pp. 317–328.

[15] Y. Xiao, P. Savolainen, A. Karppanen, M. Siekkinen, and A. Yla-Jaaski, "Practical power modeling of data transmission over 802.11g for wireless applications," in *Proc. ACM e-Energy*, 2010, pp. 75–84.

[16] Y. Xiao, M. Siekkinen, A. Wang, Y. Cui, P. Savolainen, L. Yang, S. Tarkoma, and A. Yla-Jaaski, "Modeling energy consumption of data transmission over Wi-Fi," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1760 – 1773, August 2014.

[17] L. Sun, R. K. Sheshadri, W. Zheng, and D. Koutsonikolas, "Modeling WiFi active power/energy consumption in smartphones," in *Proc. IEEE ICDCS*, 2014, pp. 41–51.

[18] N. Ding, D. Wagner, X. Chen, Y. C. Hu, and A. Rice, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," in *Proc. ACM SIGMETRICS*, 2013, pp. 29–40.

[19] M. Dong and L. Zhong, "Self-constructive high-rate system energy modeling for battery-powered mobile systems," in *Proc. ACM MobiSys*, 2011, pp. 335–348.

[20] F. Xu, Y. Liu, Q. Li, and Y. Zhang, "V-edge: fast self-constructive power modeling of smartphones based on battery voltage dynamics," in *Proc. USENIX NSDI*, 2013, pp. 43–55.

[21] D. Shin, K. Kim, N. Chang, W. Lee, Y. Wang, Q. Xie, and M. Pedram, "Online estimation of the remaining energy capacity in mobile systems considering system-wide power consumption and battery characteristics," in *Proc. IEEE ASP-DAC*, 2013.

[22] J. Lee, Y. Chon, and H. Cha, "Evaluating battery aging on mobile devices," in *Proc. ACM/EDAC/IEEE DAC*, 2015.

[23] X. Li, M. Dong, Z. Ma, and F. Fernandes, "GreenTube: power optimization for mobile video streaming via dynamic cache management," in *Proc. ACM Multimedia*, 2012.

[24] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "Using crowd-sourced viewing statistics to save energy in wireless video streaming," in *Proc. ACM Mobicom*, 2013.

[25] J. Chen, A. Ghosh, J. Magutt, and M. Chiang, "QAVA: quota aware video adaptation," in *Proc. ACM CoNEXT*, 2012.

[26] W. Lee, J. Koo, S. Jin, and S. Choi, "EQ-Video: energy and quota-aware video playback time maximization for smartphones," *IEEE Commun. Lett.*, vol. 19, no. 6, pp. 1045–1048, Jun. 2015.

[27] W. Hu and G. Cao, "Energy-aware video streaming on smartphones," in *Proc. IEEE INFOCOM*, 2015.

[28] D. H. Bui, K. Lee, S. Oh, I. Shin, H. Shin, H. Woo, and D. Ban, "GreenBag: energy-efficient bandwidth aggregation for real-time streaming in heterogeneous mobile wireless networks," in *Proc. IEEE RTSS*, 2013.

[29] Y. Go, O. C. Kwon, and H. Song, "An energy-efficient HTTP adaptive video streaming with networking cost constraint over heterogeneous wireless networks," *IEEE Trans. on Multimedia*, vol. 17, no. 9, pp. 1646–1657, 2015.

[30] "Gpac," https://gpac.wp.mines-telecom.fr, 2016.

[31] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[32] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: evidence from a large video streaming service," in *Proc. ACM SIGCOMM*, 2014.

[33] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: near-optimal bitrate adaptation for online videos," in *Proc. IEEE INFOCOM*, 2016.

[34] "ExoPlayer," http://google.github.io/ExoPlayer/, 2016.

[35] W. Lee, J. Koo, S. Choi, and Y. Park, "ESPA: Energy, usage ($), and performance-aware LTE-WiFi adaptive activation scheme for smartphones," in *Proc. IEEE WoWMoM*, 2014, pp. 1–9.

[36] O. C. Kwon, Y. Go, Y. Park, and H. Song, "MPMTP: Multipath multimedia transport protocol using systematic Raptor codes over wireless networks," *IEEE Transactions on Mobile Computing*, 2014.

[37] Q. Peng, M. Chen, A. Walid, and S. Low, "Energy efficient multipath TCP for mobile devices," in *Proc. ACM MobiHoc*, 2014, pp. 257–266.

[38] Y.-S. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, and R. J. Gibbens, "How green is multipath TCP for mobile devices?" in *Proc. ACM AllThingsCellular*, 2014, pp. 3–8.

[39] M. Lauridsen, P. Mogensen, and L. Noël, "Empirical LTE smartphone power model with DRX operation for system level simulations," in *Proc. IEEE VTC*, 2013, pp. 1–6.

[40] N. Zhang, P. Ramanathan, K.-H. Kim, and S. Banerjee, "Powervisor: a battery virtualization scheme for smartphones," in *Proc. ACM MCS*, 2012.

[41] D. Ferreira, E. Ferreira, J. Goncalves, V. Kostakos, and A. K. Dey, "Revisiting human-battery interaction with an interactive battery interface," in *Proc. ACM UbiComp*, 2013.

[42] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "AppScope: Application energy metering framework for Android smartphone using kernel activity monitoring," in *Proc. USENIX ATC*, 2012, pp. 387–400.

[43] J. Koo, W. Lee, Y. Park, and S. Choi, "PIMM: packet interval-based power modeling of multiple network interface-activated smartphones," in *Proc. ACM e-Energy*, 2015.

[44] "Maxim integrated battery management," http://www.maximintegrated.com/en/products/power/battery-management.html.

[45] "Smart battery," http://smartbattery.org.

[46] S. Gurun and C. Krintz, "A run-time, feedback-based energy estimation model for embedded devices," in *Proc. IEEE/ACM/IFIP CODES+ISSS*, 2006.

[47] J. Kim and B.-H. Cho, "State-of-charge estimation and state-of-health prediction of a Li-ion degraded battery based on an EKF combined with a per-unit system," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 9, pp. 4249–4260, 2011.

[48] W. van Schalkwijk and B. Scrosati, *Advances in lithium-ion batteries*. Springer, 2002.

[49] M. Broussely, P. Biensan, F. Bonhomme, P. Blanchard, S. Herreyre, K. Nechev, and R. Staniewicz, "Main aging mechanisms in Li ion batteries," *Elsevier Journal of power sources*, vol. 146, no. 1, pp. 90–96, 2005.

[50] B. Pattipati, B. Balasingam, G. Avvari, K. Pattipati, and Y. Bar-Shalom, "Open circuit voltage characterization of lithium-ion batteries," *Elsevier Journal of Power Sources*, vol. 269, pp. 317–333, 2014.

[51] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, "Demystifying 802.11n power consumption," in *Proc. USENIX HotPower*.

[52] National Instruments USB-6210. http://www.ni.com/data-acquisition/multifunction/.

[53] "MX player," https://play.google.com/store/apps/details?id=com.mxtech.videoplayer.ad&hl=ko.

[54] "Galaxy S5 Download Booster," http://galaxys5guide.com/samsung-galaxy-s5-features-explained/galaxy-s5-download-booster (Accessed: 18 Oct. 2016), 2016.

[55] "Airplug," http://www.airplug.com/solution.php (Accessed: 18 Oct. 2016), 2016.

[56] X. Zhu, P. Agrawal, J. P. Singh, T. Alpcan, and B. Girod, "Distributed rate allocation policies for multihomed video streaming over heterogeneous access networks," *IEEE Trans. Multimedia*, vol. 11, no. 4, pp. 752–764, Jun. 2009.

[57] J. Wu, B. Cheng, C. Yuen, Y. Shang, and J. Chen, "Distortion-aware concurrent multipath transfer for mobile video streaming in heterogeneous wireless networks," *IEEE Trans. Mobile Comput.*, vol. pp, no. 99, pp. 2607–2620, Jul. 2014.

[58] J. Wu, J. Yang, Y. Shang, B. Cheng, and J. Chen, "SPMLD: Sub-packet based multipath load distribution for real-time multimedia traffic," *J. Commun. Netw.*, vol. 16, no. 5, pp. 548–558, Oct. 2014.

[59] C. Xu, T. Liu, J. Guan, H. Zhang, and G.-M. Muntean, "CMT-QA: Quality-aware adaptive concurrent multipath data transfer in heterogeneous wireless networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2193–2205, Nov. 2013.

[60] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen, "Streaming high-quality mobile video with multipath TCP in heterogeneous wireless networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 9, pp. 2345–2361, 2016.

[61] S. Dimatteo, P. Hui, B. Han, and V. O. Li, "Cellular traffic offloading through WiFi networks," in *IEEE MASS*, 2011.

[62] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.

[63] Q. Song and A. Jamalipour, "A network selection mechanism for next generation networks," in *Proc. IEEE ICC*, 2005.

[64] Q.-T. Nguyen-Vuong, N. Agoulmine, and Y. Ghamri-Doudane, "A user-centric and context-aware solution to interface management and access network selection in heterogeneous wireless environments," *Computer Networks*, vol. 52, no. 18, pp. 3358–3372, 2008.

[65] "RFC 7233 (HTTP/1.1 range requests)," https://tools.ietf.org/html/rfc7233, 2014.

[66] Z. Yan and C. W. Chen, "RnB: rate and brightness adaptation for rate-distortion-energy tradeoff in HTTP adaptive streaming over mobile devices," in *Proc. ACM MobiCom*, 2016.

[67] M. J. Neely, "Dynamic optimization and learning for renewal systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 32–46, 2013.

[68] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *Proc. ACM SIGCOMM*, 2011.

[69] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for Internet video," in *Proc. ACM SIGCOMM*, 2013.

초 록

최신 스마트폰은 LTE와 Wi-Fi와 같은 네트워크 인터페이스 여러 개를 동시에 사용하여 전송율을 증가시키거나 네트워크 접속성을 향상시킬 수 있다. 이런 경우에, 스마트폰의 에너지 소모는 LTE와 Wi-Fi를 동시에 사용함에 따라 증가할 수 있다. 특히, 제한된 용량을 가지는 배터리로 동작을 하는 스마트폰에서 에너지 소모는 중요한 이슈이기 때문에, 에너지 증가와 성능 향상 사이의 trade-off를 고려하는 것이 요구된다. 에너지 인지 기술과 함께 스마트폰의 LTE와 Wi-Fi 인터페이스를 전략적으로 잘 사용함으로써, 배터리 에너지 소모를 줄임과 동시에 스마트폰 어플리케이션의 성능을 최적화 하는 것이 가능해진다.

본 논문에서는 LTE와 Wi-Fi 링크를 동시에 활용할 수 있는 스마트폰에서 에너지 인지를 가능하게 하는 다음 세 가지 전략을 고려하였다. (i) LTE와 Wi-Fi를 동시에 사용하는 스마트폰의 전력 소모 모델링, (ii) 스마트폰의 배터리 소모율의 실시간 예측 기법, (iii) dynamic adaptive streaming over HTTP (DASH) 기반의 비디오 스트리밍의 성능 최적화.

먼저, 동시에 여러 네트워크를 활성화하여 사용하는 스마트폰의 정밀한 전력 소모 모델링을 제시한다. 패킷 처리에 의해 소모되는 전력과 네트워크 인터페이스에서 소모되는 전력을 분해하여, 다중 네트워크 인터페이스를 활성화시키는 경우의 정밀한 전력 소모 모델을 구성한다. 다양한 시나리오에서 측정된 전력과 모델을 통해 예측된 전력을 비교하며 제안하는 전력 소모 모델의 정확성을 평가하였다. 기존 전력 소모 모델에 비해 단일 네트워크 통신의 경우에도 $7\% - 35\%$ 만큼 추정 오차를 줄였으며, 다중 네트워크 통신의 경우 추정 오차를 25% 줄임을 보였다.

두 번째로, 스마트폰의 실시간 에너지 인지를 가능하게 하기 위해서, 스마트폰에서 사용하고 있는 Li-ion 배터리의 특성을 고려하여 배터리 소모율을 추정하는 기법을 고안한다. Li-ion 배터리는 온도와 노화 상태에 따라서 가용 용량과 내부 저항이 달라지기 때문에, 온도와 노화 상태에 따라서 배터리 소모율이 달라질 수 있다. 배터리 특성을 모델링하는 것은 어려운 일이기 때문에, effective resistance 개념을 도입하여 용량과 내부 저항을 모르고도 배터리 소모율을 예측할 수 있는 *BattTracker*를 고안한다. *BattTracker*는 실시간 배터리 소모율을 최대 0.5초 마다 추정할 수 있다. 실제 스마트폰으로 다양한 실험을 통하여 *BattTracker*가 배터리 소모 예측을 5% 오차율 이내로 예측함을 보였으며, 이를 활용하면 높은 시간 해상도로 스마트폰의 에너지 인지 동작이 가능해진다.

마지막으로, 애너지 인지 기법과 LTE와 Wi-Fi 링크를 동시에 활용하는 것을 dynamic adaptive streaming over HTTP (DASH) 기반 비디오 스트리밍 어플리케이션에 적용한다. 스마트폰에서 LTE와 Wi-Fi 링크를 동시에 활용하는 것은 다양한 측면에서 DASH 기반의 비디오 스트리밍의 성능을 향상시킬 수 있다. 그러나, 배터리 에너지와 LTE 데이터 사용량을 절약하면서 끊김없는 고화질의 비디오를 스트리밍하는 것은 도전적인 일이다. 따라서, DASH 비디오를 시청하는 사용자의 Quality of Experience (QoE)를 향상시키기 위하여 LTE와 Wi-Fi를 동시에 활용하는 DASH video chunk 요청 기법인 REQUEST를 제안한다. REQUEST는 주어진 배터리 에너지와 LTE 사용량 예산 내에서 최적에 가까운 품질의 비디오를 끊김없이 스트리밍해 주는 것을 가능하게 한다. 다양한 환경에서의 시뮬레이션과 실 환경에서의 측정을 통하여, REQUEST가 기존 비디오 스트리밍 기법에 비해 평균 비디오 품질, 재버퍼링, 자원 낭비량 관점에서 상당히 우수함을 보인다.

요약하자면, 우리는 LTE와 Wi-Fi의 동시 사용하는 스마트폰에서의 전력 모델링 방법론, 실시간 배터리 소모율 추정 기법, DASH 기반 비디오 스트리밍 성능 최적화를 제안한다. 이 연구를 통해서, 우리는 프로토타입 구현과 실험 장비들을 통한 실측 기반으로 LTE와 Wi-Fi를 동시에 활용하는 스마트폰을 위한 에너지 인지 기술을 제안한다. 제안하는 기법들의 성능은 상용 스마트폰에 구현하여 실 환경에서의 성능 평가를 통해 검증하였다.

주요어: Wi-Fi, LTE, 스마트폰, 전력 모델링, 리튬 이온 배터리, 에너지 인지, HTTP에서의 적응적 비디오 스트리밍 (DASH).
학번: 2011-20785